Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Nachfolger: Commodore 16

Abenteuerspiel Valhalla

Spannung und Widerstand

Befehlsstrukturen

Der gesteuerte Griff

Heft 23

Ein wöchentliches Sammelwerk

computer kurs

Heft 23

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

Computer Kurs ist ein wöchentlich erscheinendes Sammelwerk. Die Gesamtzahl der Hefte ergibt ein vollständiges Computer-Nachschlagewerk. Damit Sie jede Woche Ihr Heft erhalten, bitten Sie Ihren Zeitschriftenhändler, Computer Kurs für Sie zu reservieren.

Zurückliegende Hefte
Ihr Zeitschriftenhändler besorgt Ihnen gerne zurückliegende Hefte. Sie können sie aber auch direkt beim Verlag bestellen.

Deutschland: Das einzelne Heft kostet DM 3,80. Bitte füllen Sie eine Postzahlkarte aus an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Computer Kurs

Österreich: Das einzelne Heft kostet öS 30. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs, Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Computer Kurs.

Schweiz: Das einzelne Heft kostet sfr 3,80. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

Abonnement

Sie können Computer Kurs auch alle 2 Wochen (je 2 Ausgaben) per Post zum gleichen Preis im Abonnement beziehen. Der Abopreis für 12 Ausgaben beträgt DM 45,60 inkl. Mwst., den wir Ihnen nach Eingang der Bestellung berechnen. Bitte senden Sie Ihre Bestellung an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk Service, Postgiroamt Hamburg 86853-201, Postfach 105703, 2000 Hamburg 1, Kennwort: Abo Computer Kurs. Bitte geben Sie an, ab welcher Nummer das Abo beginnen soll und ob Sie regelmäßig für jeweils 12 Folgen einen Sammelordner wünschen. Bei Bestellungen aus Österreich oder Schweiz senden Sie Ihren Auftrag bitte auch an die Hamburger Adresse. Berechnung und Zahlung erfolgen in Landeswährung zum Ladenpreis.

WICHTIG: Bei Ihren Bestellungen muß der linke Abschnitt der Zahlkarte Ihre vollständige Adresse enthalten, damit Sie die Hefte schnell und sicher erhalten. Überweisen Sie durch Ihre Bank, so muß die Überweisungskopie Ihre vollständige Anschrift gut leserlich enthalten.

SAMMELORDNER
Sie können die Sammelordner entweder direkt bei Ihrem Zeitschriftenhändler kaufen (falls nicht vorrätig, bestellt er sie gerne für Sie) oder aber Sie bestellen die Sammelordner für den gleichen Preis beim Verlag wie folgt:

Deutschland: Der Sammelordner kostet DM 12. Bitte füllen Sie eine Zahlkarte aus an: Marshall Cavendish International Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Sammelordner Computer Kurs.

Österreich: Der Sammelordner kostet öS 98. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Sammelordner Computer Kurs

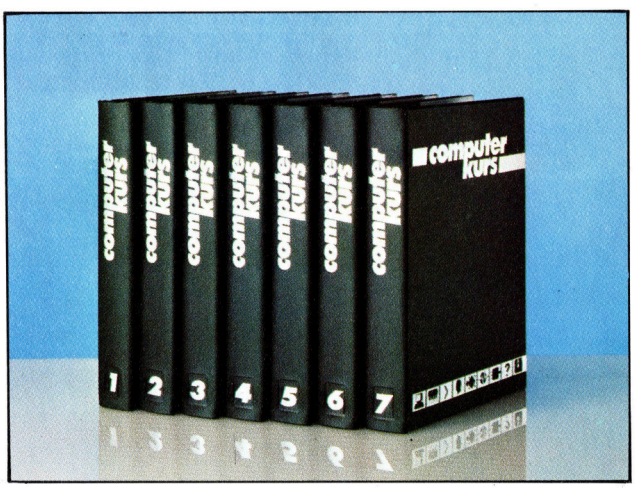
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

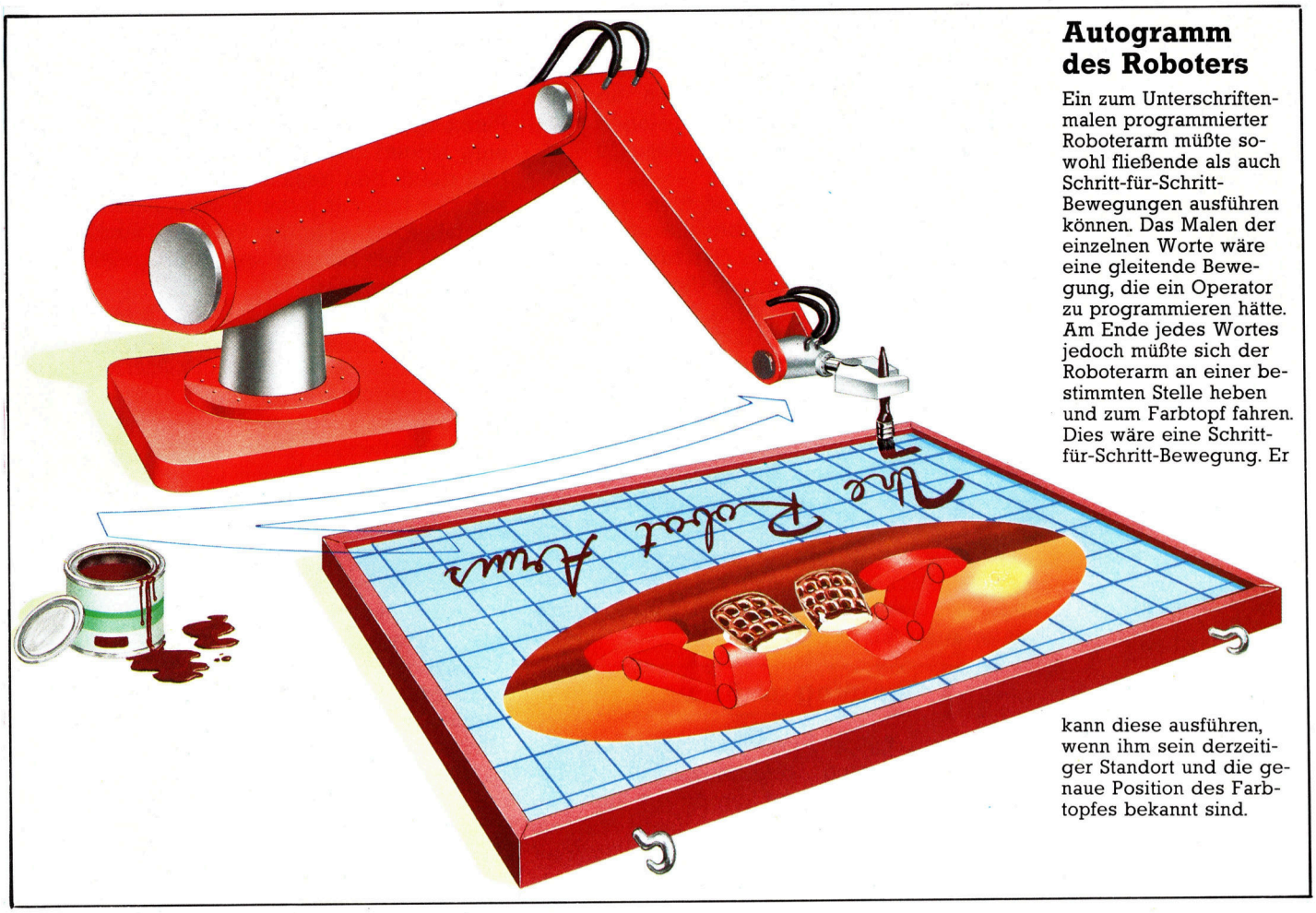
Alle 12 Hefte erscheint ein Teilindex. Die letzte Ausgabe von Computer Kurs enthält den Gesamtindex — darin einbezogen sind Kreuzverweise auf die Artikel, die mit dem gesuchten Stichwort in Verbindung stehen.

Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Joachim Seidel, Elke Leibinger, Susanne Brandt, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1, Tel.: 040/23 40 85



Druck: E. Schwend GmbH, Schmollerstraße 31, 7170 Schwäbisch Hall



Autogramm des Roboters

Ein zum Unterschriftenmalen programmierter Roboterarm müßte sowohl fließende als auch Schritt-für-Schritt-Bewegungen ausführen können. Das Malen der einzelnen Worte wäre eine gleitende Bewegung, die ein Operator zu programmieren hätte. Am Ende jedes Wortes jedoch müßte sich der Roboterarm an einer bestimmten Stelle heben und zum Farbtopf fahren. Dies wäre eine Schritt-für-Schritt-Bewegung. Er kann diese ausführen, wenn ihm sein derzeitiger Standort und die genaue Position des Farbtopfes bekannt sind.

Bewegungsabläufe

Es wird aufgezeigt, welche Mittel es gibt, einen Roboter so zu programmieren, daß er „intelligente" Aufgaben ausführen kann.

Es ist einleuchtend, daß Roboterarme ähnlich wie menschliche Gliedmaßen konstruiert werden müßten: ein Skelett, um dem Ganzen Halt zu geben, und „Muskeln", um Kraft ausüben zu können. Doch für die Ausführung einer Aufgabe ist „Intelligenz" für den Arm erforderlich.

Auf den ersten Blick scheint der Gedanke eines „intelligenten Armes" unsinnig. Nehmen wir zum Verständnis jedoch ein einfaches Beispiel. Stellen Sie sich vor, Sie säßen vor einem Tisch, der bis auf einen kleinen Gegenstand auf der linken Seite der Tischplatte leer ist. Ihre Aufgabe bestünde nun darin, diesen Gegenstand von der linken Seite des Tisches zu seiner rechten zu befördern. Zwei Arten der Intelligenz spielen bei diesem Vorgang eine Rolle. Zunächst erfolgt die Wahrnehmung sowohl des Tisches als auch des Gegenstandes. Daraufhin ist die Entscheidung zu treffen, daß das Objekt von der einen Seite zur anderen zu bewegen ist. Dieser Vorgang ist mit weiteren „Bausteinen" verknüpft, die wir als „Intention" und „zielorientiertes Verhalten" bezeichnen. Die für den Roboterarm benötigte Art von Intelligenz findet auf einer sehr niedrigen Ebene statt; denn die Entscheidung über die Bewegung von Arm und Hand ist bereits getroffen worden. Es geht also lediglich darum, die Hand in die richtige Position zu bringen und sicherzustellen, daß die Hand das Objekt greift und zum richtigen Zeitpunkt wieder losläßt.

Das mag einfach und leicht klingen. Doch wer bezweifelt, daß es sich hierbei um einen intelligenten Vorgang handelt, sollte sich einmal ansehen, was geschieht, wenn ein kleines Kind versucht, das zuvor Beschriebene zu tun. Dem Kind wird es zuerst mißlingen, den Gegenstand zu fassen. Es wird ihn mehrfach an der falschen Stelle absetzen und bei diesen Versuchen generell einen unsicheren Eindruck machen. Das Kind versucht nämlich, seine Intelligenz zu trainieren, um Arme und Hände in einer wenig vertrauten, dreidimen-

Um sich von einem Punkt zum anderen bewegen zu können, muß der Zweipunkt-Roboterarm eine Bewegung um seinen Drehpunkt (Winkel R) machen und sowohl die Winkel der Schulter (S) als auch die der Ellenbogen (E) verändern. Sind die cartesischen Koordinaten des Zielpunktes (X2, Y2, Z2), wird die notwendige Veränderung so berechnet:

```
A1=SQR(X1^2 + Y1^2
   + Z1^2)
A2=SQR(X2^2 + Y2^2
   + Z2^2)
```

Drehpunkt:

```
R1=ARCTAN(Y1/X1)
R2=ARCTAN(Y2/X2)
Veränderung=(R2—R1)
```

Schulter:

```
S1=ARCCOS(Z1/A1)
   +ARCCOS
   ((A2^2+U^2-L^2)/
   2*A2*U))
S2=ARCCOS(Z2/A2)
   +ARCCOS
   ((A2^2+U^2—L^2)
   (2*A2*U))
Veränderung=(S2—S1)
```

Ellenbogen:

```
E1=ARCCOS((U^2+L^2-
   A1^2)/(2*U*L))
E2=ARCCOS
   ((U^2+L^2-A2^2)/
   (2*U*L))
Veränderung=(E1—E2)
```

U und L bezeichnen die Länge des oberen bzw. unteren Arms.

sionalen Welt richtig zu bewegen. Hat es dies erst einmal gelernt, scheinen die Bewegungen immer automatischer zu werden. Bewußtes Denken ist nicht mehr erforderlich, und wir würden das Bewegen des Gegenstands auch nicht mehr als „Intelligenz erfordernde Aufgabe" betrachten.

Der Roboterarm befindet sich in der gleichen Situation. Er verfügt über die rein physischen Voraussetzungen zur Durchführung von Aufgaben, muß aber lernen, diese „automatisch" auszuführen. Die einfachste Methode, einem Arm bestimmte Aufgaben beizubringen, ist, ihn durch eine Bewegungsabfolge zu führen. Vornehmlich bei Industrierobotern kommt dieses Verfahren zum Einsatz. Der „Lehrer" nimmt den Roboter im wörtlichen Sinne an die Hand und verdeutlicht ihm dabei die einzelnen zu befolgenden Schritte. Das hat den großen Vorteil, daß die betreffende Person nicht wissen muß, wie der Roboter rein technisch funktioniert. Sie muß lediglich die Abfolge der Bewegungen kennen, die der Arm später auszuführen hat.

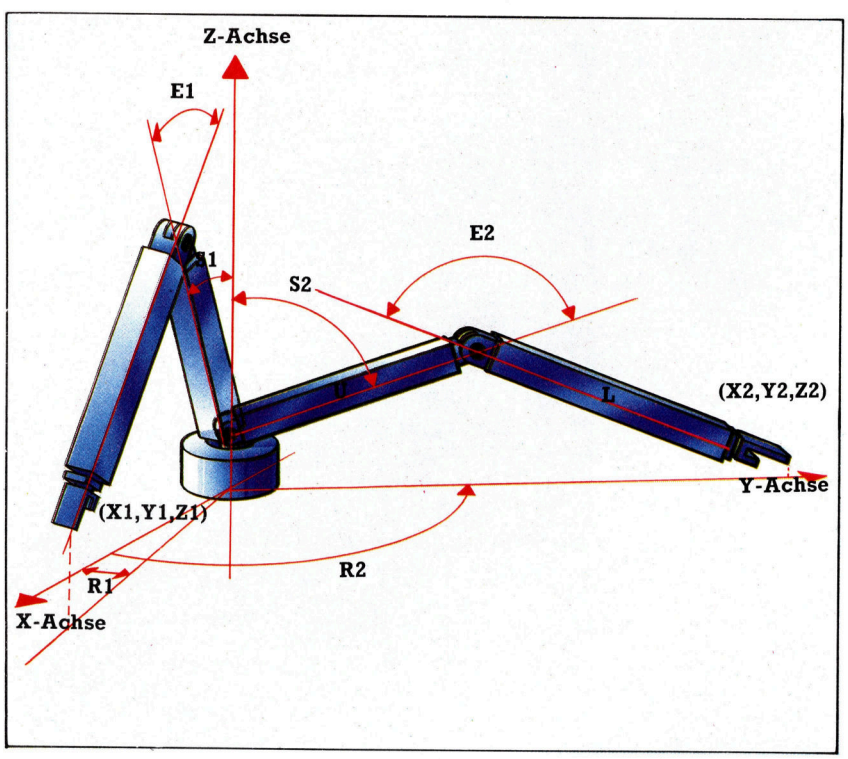
Training muß sein

Zwei „Ausbildungs"-Arten finden bei Roboterarmen Anwendung: schrittweise und gleitende. Beim schrittweisen Lernprozeß führt der Operator den Arm in eine bestimmte Position und drückt dann einen Knopf, um dem Roboter zu signalisieren, daß er sich an diese Position „zu erinnern" hat. Dann wird der Arm auf die nächste Position gebracht und das Signal wird wieder ausgelöst. Diese Sequenz wird so lange fortgesetzt, bis sämtliche Etappen im Speicher des Roboters abgelegt sind. Nach Beendigung dieser Ausbildungsstufe kann der Roboter in den „Ausführungs"-Modus geschaltet werden und bewegt sich nun genau in der erlernten Form Schritt um Schritt von einem Punkt zum nächsten. Beim „gleitenden" Training steuert der Operator den Roboter ohne Halt durch den gesamten Arbeitsgang, wobei sich der Roboter jeder Einzelposition des Gesamtablaufes erinnert. Bei der Durchführung vollzieht der Roboter dann die gelernte Abfolge genauso.

Nun ist zu überlegen, wie sich ein Roboter an die Bewegungsabläufe „erinnern" kann. Im Trainingsmodus speichert der Roboter mittels seiner internen Sensoren die jeweils erreichten Positionen. Das erfolgt z. B. mit Hilfe der Achsenzählwerke, über die wir an anderer Stelle bereits berichteten. Die jeweils erreichte Position wird also ebenso registriert wie die vollzogene Bewegung. Dies geschieht entweder durch direkte Eingabe oder aber durch dauerhafte Speicherung auf Band oder Diskette. Im Ausführungsmodus ruft der Roboter alle relevanten Daten ab und verwandelt sie wiederum in Bewegungen.

Es mag überraschen, daß es dem Roboter leichter fällt, sich der durch „gleitendes" Training vermittelten Schrittfolge zu „erinnern" und danach zu arbeiten. Das hängt damit zusammen, daß der Arm lediglich einer exakt bestimmten Strecke zu folgen hat. Dafür sind jedoch sehr viele einzelne Daten erforderlich. Eine Strecke wird nämlich oft von mehreren tausend Einzelpunkten definiert im Vergleich zu den wenigen Positionen, die bei der „schrittweisen" Programmierung eingegeben werden. Eine zweite Schwierigkeit resultiert aus der Tatsache, daß der Arm die Bewegung glatt und exakt ausführen soll.

Wenn alle Achsen simultan bewegt bzw. gesteuert werden müssen, so müßte beispielsweise ein Farbsprühroboter seine drei Armgelenke und außerdem noch die drei Handgelenke bewegen, um genau sprühen zu können. Entweder müßten die Daten daher in der CPU sehr schnell verarbeitet werden, oder die einzelnen Gelenkteile des Roboters müßten mit insgesamt sechs einzelnen Prozessoren ausgestattet werden.



Berechnung des Weges

Der schrittweise arbeitende Roboter hat eine schwere Aufgabe, da er, obwohl er „weiß", wohin er sich bewegen soll, nicht „gelernt" hat, wie er dorthin gelangt. Nun könnte er die einzelnen Gelenke so lange bewegen, bis die erforderliche Position erreicht ist. Doch das wäre eine Vergeudung von Zeit und Energie. Es wäre sinnvoller, wenn der Roboter den kürzesten Weg von einem Punkt zum anderen unter Berücksichtigung seiner eigenen Position berechnen würde. Die dafür erforderlichen Berechnungen sind allerdings sehr kompliziert,

da zur Bewegung der Hand in einer Geraden zwischen zwei definierten Punkten cartesische Koordinaten verwendet werden müssen, wogegen die Position der Arme selbst mit einem anderen Koordinatensystem zu definieren ist. Der Roboter muß also, um effektiv arbeiten zu können, verschiedene geometrische Probleme auf einmal lösen. Bei Industrierobotern, die ja Gegenstände zu bewegen haben, die zuweilen einige hundert Kilo schwer sind, ist das Sparen von Zeit und Energie durch Wahl des kürzesten Weges ein wichtiger Gesichtspunkt.

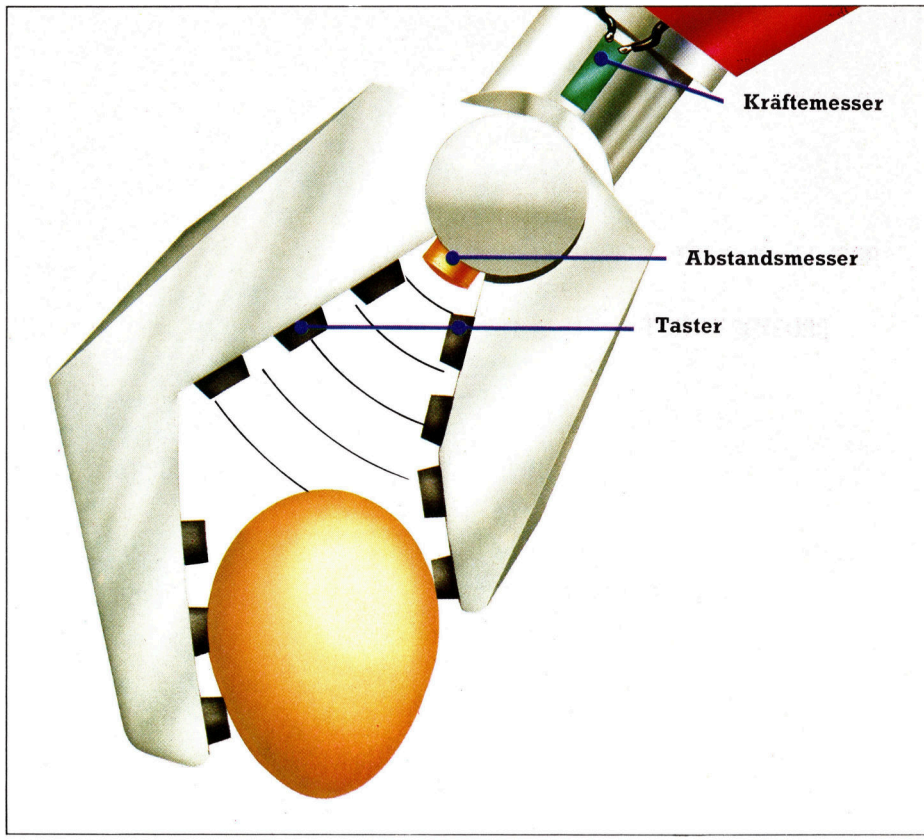
Optimale Geschwindigkeit

Ein weiteres Problem, das sich dem schrittweise arbeitenden Roboter stellt, ist das der Eigendynamik des Arms. Wenn Sie Ihren eigenen Arm bewegen, um einen Gegenstand zu greifen, stellen Sie fest, daß er sich zunächst langsam aus seiner Ausgangsposition bewegt, dann an Geschwindigkeit zunimmt, Höchstgeschwindigkeit erreicht und schließlich in der gewünschten Endposition verhält. Die Vorteile eines ebenso arbeitenden Roboterarms liegen auf der Hand. Viele solcher Arme arbeiten jedoch mit gleichbleibender Geschwindigkeit, erreichen fast augenblicklich das Höchsttempo und beenden abrupt die Bewegung nach Erreichen des Endpunktes. Damit wird der Arm einer größeren Belastung ausgesetzt als einer, dessen Bewegung in jeder Phase glatt verläuft. Der Roboter muß also nicht nur den idealen Weg finden, sondern auch die optimale Geschwindigkeit errechnen.

Selbst wenn der Arm so programmiert ist, daß er die Bewegungen präzise nachvollzieht, kann sich bei der Durchführung zeigen, daß die Bewegung nicht so vollzogen wird wie gewünscht. Man muß also einen Weg finden, den Bewegungsablauf wie gewünscht editieren zu können. Möglich ist das, indem man die Bewegungen zunächst in Listenform erfaßt und so jede Position einzeln gespeichert hat, gefolgt von der Adresse, wo die nächste Position zu finden ist. Sind Veränderungen erforderlich, wird der Arm lediglich in die korrekturbedürftige Position gebracht, dort gestoppt und ein neuer Bewegungsablauf eingefügt.

Neben dieser Methode ist eine weitere üblich, um Arme intelligenter zu machen. Sie besteht darin, eine Reihe programmierter Anweisungen in einem Computer zu speichern. Normalerweise wird jeder Roboter individuell programmiert, und zur Steuerung seiner Bewegungen wird eine eigene Programmiersprache eingesetzt. Erforderlich aber wäre eine Eingabeform, die es dem Programmierer ermöglicht, LOGO-ähnliche Befehle zu verwenden, mit denen Bewegungen in den drei Dimensionen spezifiziert werden können.

Die Problematik ist der ähnlich, die beim „schrittweisen" Training dargelegt wurde, da viele Faktoren zu berücksichtigen sind. Soll sich beispielsweise ein Roboterarm um zehn Einheiten vorwärts bewegen, müßte die Schulterhaltung geändert werden, damit der Arm weiter ausgreifen kann. Dies aber beinhaltet eine bogenförmige Aufwärtsbewegung, die ihrerseits wieder durch eine entsprechende abwärtige Gegenbewegung des Ellenbogens korrigiert werden müßte. Daran ist deutlich zu sehen, daß die Anweisungen für eine ganz einfache Bewegung in zwei verschiedene Gruppen von Instruktionen übertragen werden müssen, da in der Regel zwei separate Gelenke betroffen sind.



Das Aufnehmen eines Eies dient zur Kontrolle der Sensoren eines Roboterarms und seiner Feedback (Rückkoppelungs)-Kontrollmechanismen. Der Abstandsmesser im Greifer muß feststellen, daß das Ei nah genug für den Zugriff ist. Darauf beginnen die Finger, sich zu schließen, bis die Tastsensoren Kontakt mit dem Ei melden. Diese Information wird nun mit denen des Abstandsmessers verglichen, die Finger schließen sich soweit erforderlich, und der Arm hebt sich. Ein plötzliches Absinken des Abstandswertes würde bedeuten, daß das Ei rutscht. Also müssen die Finger noch dichter zueinander geführt werden, bis eine vorgegebene Kraftgrenze erreicht ist oder aber ein Sinken des Kraftwertes signalisiert, daß die Eierschale zerbrochen ist.

Die „intelligente" Hand

Andere Probleme können sich ergeben, wenn der Roboter einen Gegenstand aufzunehmen hat. Für das Erkennen unterschiedlich geformter Objekte wird eine „intelligente" Hand benötigt, die die Präsenz oder Nichtpräsenz des Gegenstandes, seine Entfernung zur Hand und die von der Hand auszuübende Kraft beim Ergreifen des Gegenstandes zu kontrollieren bzw. zu fühlen hat. „In den Griff" zu bekommen sind diese Probleme, indem man die Hand mit einem Abstandsmesser sowie Tast-Sensoren für Kräftemessung ausstattet, die eine Rückmeldung an den steuernden Computer geben und diesem Korrekturen ermöglichen. Eine intelligente Hand sollte in der Lage sein, ein rohes Ei zu greifen, festzuhalten und es wieder abzulegen. Noch schwieriger wäre das Anheben einer reifen Birne. Weiche Früchte bekommen leicht Druckstellen.

Management-informationen

In unserer letzten Folge haben wir untersucht, wie Lagerbewegungen gesteuert werden. Um alle Abläufe vom Einkauf bis zum Verkauf transparent zu machen, setzt man strukturierte Codes ein.

Das Stock Recording System ist eines der Anwendungsprogramme für den Dragon 64, das unter dem Betriebssystem OS9 funktioniert. Dieses Betriebssystem wurde von Dragon aus dem UNIX-System entwickelt und enthält Möglichkeiten des Multi-Programming und Multi-Tasking.

In früheren Folgen dieser Serie haben wir verschiedene Methoden behandelt, mit denen in einer Lagerhaltung unterschiedliche Warengruppen gekennzeichnet werden: durch Kennungen, die mit den Artikelnummern verbunden sind.

Da sich das Ergebnis dieser Kennzeichnung in relativ festen Kategorien niederschlagen soll, die Datenmengen aber durch die Speicherkapazität des Computers begrenzt sind, sollte ein derartiges System ökonomisch aufgebaut sein. Es müssen ausreichende Informationen für die Geschäftsleitung bereitstehen, während gleichzeitig der Computer nicht mit überflüssigen Daten und langen Verarbeitungszeiten blockiert werden darf.

Das Dragon Data's Stock Recording System für den Dragon 64 bietet mehrere Felder, in denen die Einzelheiten der Waren erfaßt werden können: Artikelnummer, Beschreibung, Mindestbestand, Einkaufspreis, Verkaufspreis und Packungseinheit.

Diese Felder bilden die Struktur der Lagerhaltung. So ergibt sich aus der Differenz zwischen Einkaufs- und Verkaufspreis der Bruttogewinn. Die Artikel lassen sich zu Analysezwecken in Gruppen zusammenfassen. Die Maßeinheit ist wichtig, da Artikel in unterschiedlichen Packungseinheiten verkauft werden können. Bei der einen Warengruppe könnte jeder Artikel einzeln gezählt werden, bei der anderen jedoch nur die Packung (z. B. Nägel oder Schrauben).

Einer der interessantesten Aspekte beim Aufbau eines Lagersystems besteht darin, daß die Daten dynamisch sind und nicht statisch, das heißt, sie verändern sich ständig. Wir haben gesehen, daß ein Hauptbuch dagegen konstante Informationen über Kunden und Konten enthält (die Stammdaten) und auch Informationen über sämtliche Vorgänge innerhalb der Konten.

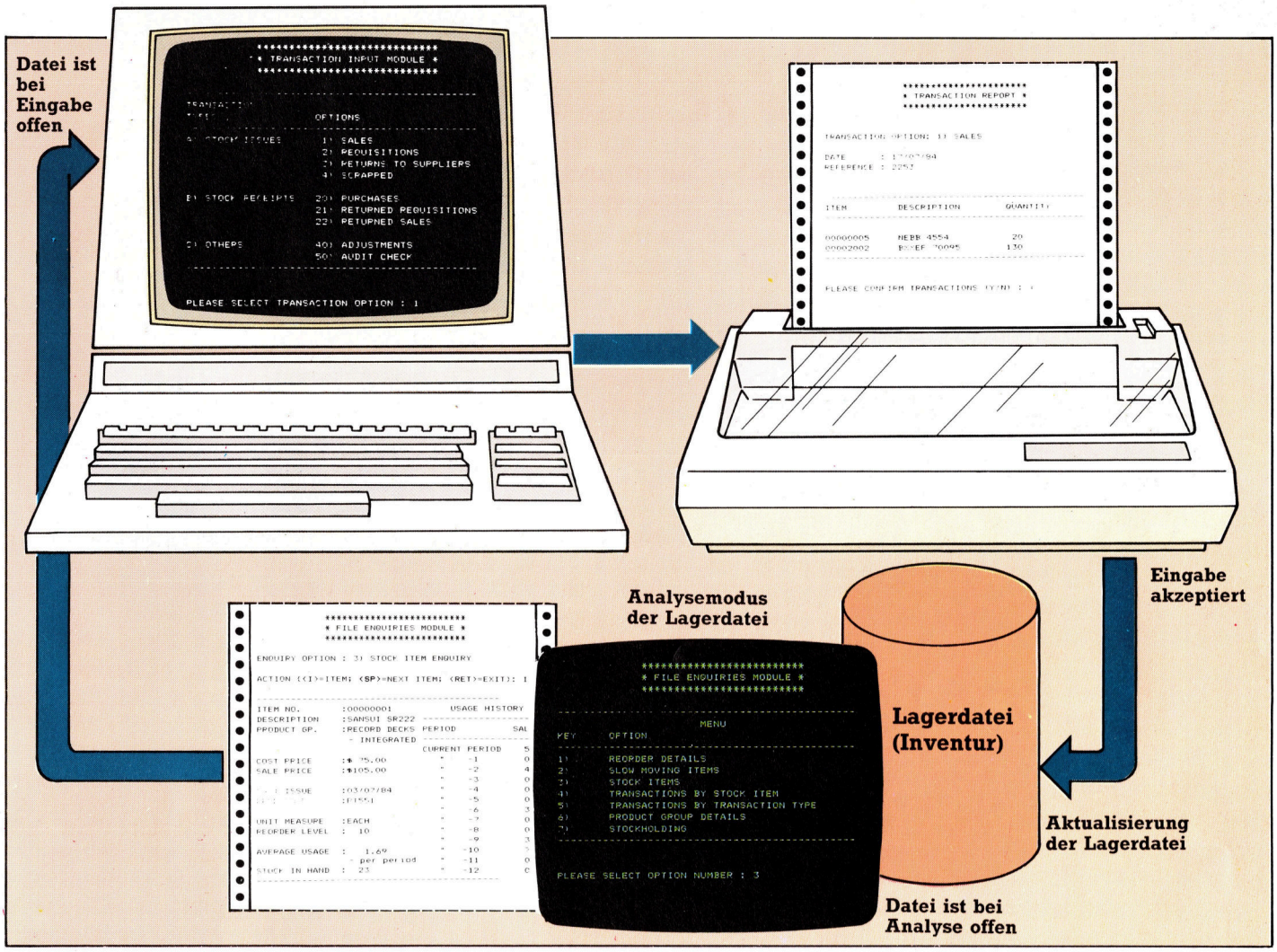
Bei einer Lagerhaltung sind die Grenzen zwischen Stamm- und Bewegungsdaten in den Datensätzen weniger klar definiert. Artikelbezeichnung, Warengruppenzuordnung und Codierung entsprechen den unveränderlichen Informationen der Kunden- oder Lieferantenstammdaten in einem Fakturiersystem. Ein Lagerhaltungspaket bietet jedoch zusätzlichen Raum für ausführliche Informationen.

In den Kundenstammdaten verändert sich die Telefonnummer oder Adresse eines Kunden nur selten. Das Programm bietet daher die Möglichkeit, diese Daten manuell zu ändern.

Daten aktualisieren

Der Einkaufs- und Verkaufspreis eines Artikels kann sich jedoch bei jeder neuen Lieferung verändern. Die Aktualisierung der Daten sollte hierbei aus Zeitgründen von dem Programm selbst vorgenommen werden (indem es die Daten der Bestellung bei der Anlieferung neuer Waren automatisch in die Stammdaten übernimmt). Natürlich besteht nach wie vor die Notwendigkeit, Stammdaten direkt durch den Anwender zu ändern, ein Großteil der neuen Daten läßt sich aber von dem System automatisch erzeugen.

Um die Systematik von Lagerprogrammen verstehen zu können, müssen wir uns deren Eingaberoutinen, Listen und Berichte ansehen. Die Abbildung zeigt die Dragon-Lagerhaltung mit den unterschiedlichen Dateien. Dabei sind die drei wichtigsten Elemente die Eingaberoutinen der Bewegungsdaten, die Bewegungsdaten selbst und die Datei, die diese speichert.



Sämtliche Bildschirmmasken dieses Systems besitzen das gleiche Eingabeformat für Bewegungsdaten, wobei sich die einzelnen Arbeitsabläufe aus der Maske selbst ergeben. Alle kommerziellen Programme dieser Art sind so „benutzerfreundlich" wie möglich aufgebaut, wobei die notwendigen Informationen zur Bearbeitung auf dem Bildschirm dargestellt werden. Dabei müssen mindestens zwei Programmanforderungen erfüllt sein.

Zunächst muß ein Programm die Fähigkeit haben, mit den Daten einiger Eingabefelder Berechnungen auszuführen. Zum anderen muß es den Anwender auch bei Korrektureingaben mit ausreichenden Informationen versorgen.

Listen und Analysen

Sind das Paßwort und die Artikelnummer eingegeben, überprüft der Computer, ob der betreffende Artikel in der Datei überhaupt existiert. Ist die Artikelnummer vorhanden, wird automatisch die zugehörige Artikelbezeichnung dargestellt. Dann erst kann die verkaufte Menge eingetragen werden.

Diese Informationen verdichtet der Computer in einer großen Anzahl von Listen und Analysen. So lassen sich z. B. mit einer Option des Hauptmenüs – FILE ENQUIRIES – sieben Untermenüs aufrufen, die Aufschluß über Lagerartikel, Lagerbewegungen und Warengruppen geben: STOCK DETAILS, SLOW MOVING ITEMS, RE-ORDER DETAILS, TRANSACTIONS (BY STOCK ITEMS), TRANSACTIONS (BY TRANSACTION TYPE), PRODUCT GROUP DETAILS und STOCKHOLDING.

Verkäufe haben Einfluß auf alle Listen. Wird etwa eine Liste der Nachbestellungen gedruckt, dann überprüft das Programm, ob die verkauften Mengen die Lagermenge eines Artikels unter den Mindestbestand gebracht hat. Ist das der Fall, muß eine ausreichende Menge nachbestellt werden.

Jede Einzelheit der Bewegungsdaten ist wichtig und wird für die Verarbeitung eingesetzt. Die verschiedenen Analysemöglichkeiten dieses Systems machen deutlich, wie viele Informationen eine Geschäftsleitung aus den Daten der Verkaufsvorgänge ziehen kann. Der Jahresüberblick bietet unter anderem Aufschluß darüber, welche Artikelmengen mit welcher Geschwindigkeit umgesetzt wurden. Bei dem Jahresabschluß werden diese Daten in zusammengefaßter Form gespeichert, um auch einen Vergleich mit den Vorjahren zu ermöglichen.

Das Lagerprogramm nimmt Eingaben über Lagerbewegungen nur an, nachdem es das Paßwort des Anwenders und die Gültigkeit des Vorgangs überprüft hat. Die Artikeldatei ist dabei eröffnet und wird automatisch aktualisiert. Das Programm enthält außerdem ein Datenbankmodul, mit dem ein Anwender Dateien untersuchen und Listen unter verschiedensten Gesichtspunkten erzeugen kann.

Die Vielseitige

Die EP-44 von Brother sieht zwar wie eine gewöhnliche Schreibmaschine aus, aber sie hat es in sich: Das Gerät ist als Drucker, Rechner und Terminal einsetzbar. Mit einem Preis von nur etwa 800 Mark bietet es sich geradezu dafür an.

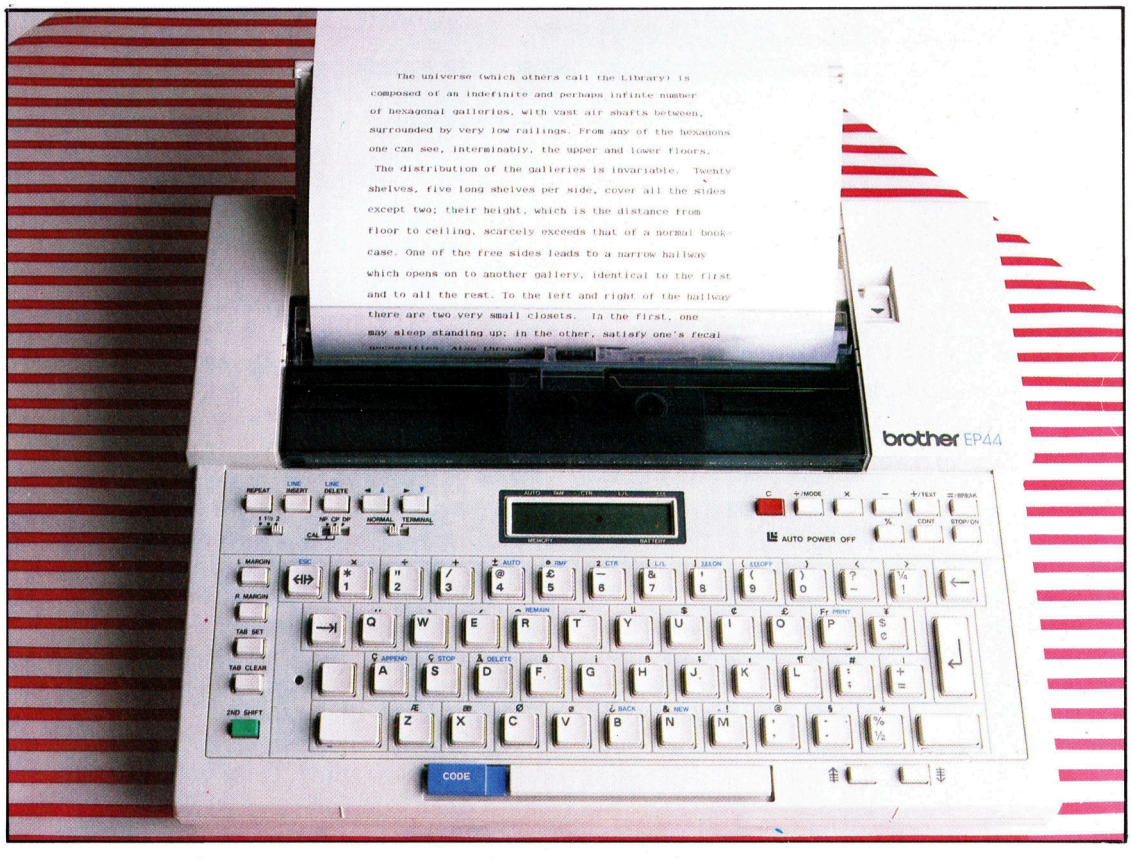
Die Brother EP-44 wiegt nur fünf Pfund und arbeitet mit Batteriebetrieb – also eine echte Reiseschreibmaschine. Für einen Aufpreis von ca. 70 Mark bekommt man auch das zugehörige Netzteil. Die Tastatur enthält außer den üblichen Schreibtasten vier Tasten für die Textverarbeitung und sieben für mathematische Operationen. Auf der LCD-Anzeige werden fünfzehn Zeichen dargestellt, so daß eine Textkorrektur vor dem Ausdrucken möglich ist.

Für normale Schreibarbeiten wird der Betriebsartenschalter der EP-44 auf „Direct Print" gestellt. Der Text erscheint dann im LCD-Fenster und wird gleichzeitig ausgedruckt. Die Tabulatoren, der Zeilenabstand und die Randbreiten werden wie bei einer konventionellen Schreibmaschine gesetzt. Der Papiertransport (wahlweise vorwärts oder rückwärts) erfolgt durch Druck auf einen der beiden Knöpfe rechts unten am Tastenfeld. Die Tastatur erreicht zwar nicht ganz den Qualitätsstandard einer Büromaschine, ist aber für normale Ansprüche völlig ausreichend.

Die EP-44 arbeitet mit einem Thermo-Druckkopf, der die Schriftzeichen als Punktmuster auf dem Papier einbrennt. Die meisten Thermodrucker verwenden dabei 9x7 Matrixpunkte – die EP-44 erreicht mit ihrer 24x18-Matrix eine merklich bessere Wiedergabe.

Der Nachteil beim Thermodruckverfahren ist die niedrige Geschwindigkeit – bei diesem Gerät unter 16 Zeichen/Sekunde. Für den reinen Schreibmaschinenbetrieb ist das ein unwesentlicher Faktor, aber bei der Verwendung als Drucker spürt man den Unterschied: Ein guter Matrixdrucker ist bis zu zehnmal so schnell. Der andere Haken sind die hohen Materialkosten bei Verwendung des speziellen Thermopapiers. Die EP-44 druckt zwar auch auf normalem Papier, jedoch leidet darunter die Qualität des Schriftbildes ein wenig.

Bei der Betriebsart „Correction Print" erscheint der eingetippte Text zunächst nur in der LCD-Anzeige und wird mit fünfzehn Anschlägen Verzögerung ausgedruckt. Dadurch schreibt es sich etwas ungewohnt, aber Sie



Die Tastatur (für Deutschland in QWERTZ ausgelegt) eignet sich auch für umfangreiche Schreibarbeiten. Außer den beiden üblichen Großbuchstaben-Umschaltern (mit Feststeller) gibt es zwei weitere Umschalttasten für Sonderzeichen und für die Systembefehle. Die Mehrfachbelegung der Tasten ermöglicht eine große Auswahl an fremdsprachlichen Schriftzeichen. Zusätzlich zur eigentlichen Schreibtastatur gibt es 21 kleinere Funktionstasten, mit denen die Rechenoperationen, der Speicherbetrieb, der Druckvorgang und der Papiervorschub gesteuert sowie die Randbreiten und Tabulatorstopps eingestellt werden.

können vor dem Drucken immer noch die letzten 15 Zeichen korrigieren, ohne „literweise" Korrekturflüssigkeit verpinseln zu müssen. Das fehlerhafte Zeichen wird im Fenster mit zwei Cursor-Steuertasten angewählt.

Außerdem gibt es noch die Betriebsart „Line-By-Line", wobei der Text erst nach Betätigung der Return-Taste gedruckt wird. Vor dem Ausdruck können Sie die Zeile beliebig korrigieren, indem Sie den Fensterbereich mit den Cursortasten über der Textzeile hin- und herschieben.

Bei Textverarbeitungssystemen wird die Eingabe automatisch formatiert. Die EP-44 beginnt im „Auto Return"-Betrieb von selbst eine neue Zeile, wenn nach Betätigung der Leertaste weniger als sechs Anschläge bis zum Zeilenende verfügbar bleiben.

Der eingebaute Rechner der EP-44 erlaubt während des Ausdrucks einfache Rechnungen auf der LCD-Anzeige. Wenn Sie z. B. eine Rechnung erstellen und zwischendurch die Mehrwertsteuer oder Rabatte berücksichtigen wollen, ermitteln Sie die Zahlen, fügen diese ein und lassen die Rechnung dann bis zum Ende ausdrucken.

4 KByte für Texte

Ähnliche Möglichkeiten bieten teilweise auch andere elektronische Schreibmaschinen, – mit Ausnahme der Textverarbeitung. Die EP-44 verfügt nämlich über einen 4-KByte-Speicher, der für den Entwurf eines dreiseitigen Briefes ausreicht. Dabei werden die normalen Schreibtasten durch Drücken der blauen „Code"-Taste mit den Befehlen für die Textverarbeitung belegt. Wenn Sie einen Text in den Speicher eingeben wollen, brauchen Sie nur die Code-Taste und die Buchstabentaste „N" („Neuer Text") zu drücken. Der Betriebsartenschalter kann dabei auf „Direct" oder „Correction Print" stehen.

Sobald der Text gespeichert ist, kann er mit Hilfe der Cursortasten bearbeitet werden. Mit diesen läßt sich das Fenster nach oben, unten, links und rechts über den Entwurf schieben, wobei ein Überprüfen des ganzen Textes sowie das Löschen, Einfügen oder Ändern von Buchstaben möglich ist.

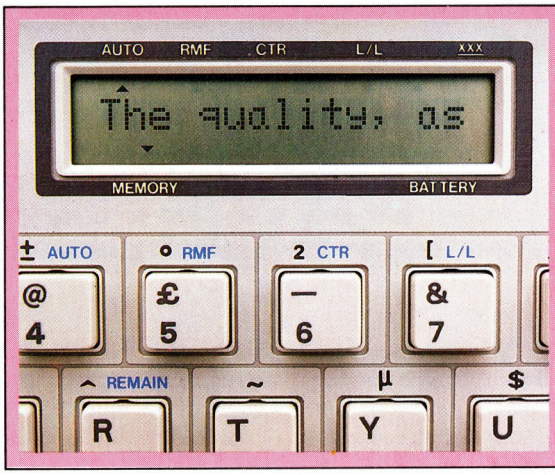
Die Texteingabe mit Speicherung ist oft sehr nützlich. – Sie können den Text variieren, ohne ihn vollständig neu eintippen zu müssen. Wenn Sie z. B. den gleichen Brief an verschiedene Empfänger schicken wollen, brauchen Sie nur im gespeicherten Text Namen und Adresse zu verändern und dann die Codetaste und gleichzeitig den Buchstaben „P" zu drükken – schon wird der geänderte Text ausgedruckt.

Durch Entfernen einer seitlichen Abdekkung am Gerät wird die RS232-Schnittstelle sichtbar, und mit dem Schalter „Terminal" verwandeln Sie das Gerät in einen Drucker für Ihren Computer. Zuvor müssen Sie jedoch die passende Baudrate, das Datenformat usw. einstellen. Das geht allerdings sehr einfach: Im Anzeigefeld laufen die verschiedenen Baudraten und andere Spezifikationen durch, und Sie betätigen den „Mode"-Schalter, wenn der korrekte Wert erscheint. Die Einstellung bleibt beim Abschalten bestehen.

Leider ist die EP-44 nicht für Endlospapier eingerichtet, so daß Einzelblatteinzug von Hand erforderlich ist. Die Schriftqualität ist aber besser als bei den meisten Matrixdruckern, so daß die EP-44 sehr gut für die Anfertigung von Briefen und anderen kurzen Schriftstücken geeignet ist.

Der erwähnte Schalter ist mit „Terminal" und nicht mit „Printer" beschriftet, weil die EP-44 nicht nur als Drucker, sondern auch als Eingabegerät zu verwenden ist. Die Maschine ist über ein Modem ans Fernsprechnetz anschließbar, so daß der Benutzer mit anderen Heimcomputer-Besitzern oder auch mit größeren Rechnern (bei geeigneter Übertragungsnorm) Verbindung aufnehmen kann.

Insgesamt verdient die Brother EP-44 Beachtung wegen ihrer Vielseitigkeit, die sie mit den vielen unterschiedlichen Funktionen auf kleinstem Raum und zu einem günstigen Preis bietet. Sie ist vor allem für Computerbesitzer interessant, die einen gleichzeitig als Schreibmaschine verwendbaren Drucker suchen.



LCD – OK?

Im LCD-Fenster der EP-44 werden bis zu 15 Schriftzeichen angezeigt, außerdem die gewählte Betriebsart und der Ladezustand der Batterien. Der im Fenster sichtbare Text kann beliebig verändert werden, was eine einfache Textverarbeitung möglich macht. Der Anzeigekontrast ist einstellbar, so daß eine Anpassung an die jeweiligen Lichtverhältnisse erfolgen kann.

Print-Out

Any smooth-finish typing paper can be used with the ribbon in place, but the paper-feed cannot accept carbons. The print quality is good on both types of paper.
Text can be
  Centred & Underlined
foreign (ÆØ¿¢µ¶), and sub- or super-scripted ($X^2=N_2+T_0$).

Diese auf der Brother EP-44 angefertigte Schriftprobe demonstriert einige der vielfältigen Darstellungsmöglichkeiten. Der Text kann zentriert sowie unterstrichen werden. Ebenso lassen sich Unter- bzw. Oberlängen darstellen.

Programmierstil

Jetzt, da wir die fundamentalen Regeln von BASIC behandelt haben, können wir uns mit dem Programmierstil sowie einigen neuen Befehlen zur Perfektionierung der Programmiertechnik befassen.

Das Adreßbuch-Programm, das wir in unserem Kursus entwickelt haben, nutzt zwar einen großen Teil der Möglicheiten der BASIC-Programmiersprache, doch längst nicht alle. An dieser Stelle werden nur einige Tips zur Verbesserung der Programmiertechnik aufgezeigt. Wir empfehlen Ihnen, zusätzlich in Ihrem Bedienungshandbuch oder der weiterführenden Literatur nachzulesen.

Bei den meisten BASIC-Versionen ist es möglich, in Maschinensprache geschriebene Routinen in ein Programm zu integrieren. Der einfachste Weg ist die Verwendung von PEEK und POKE. PEEK ist eine Anweisung, die zur Untersuchung spezieller Speicherstellen verwendet wird. Bei LET X=PEEK(1000) wird der in der Speicherstelle 1000 gespeicherte Wert der Variablen X zugewiesen. Gibt man nun die Anweisung PRINT X ein, so wird der Wert ausgegeben, der sich in der Speicherstelle 1000 befindet. Das folgende Programm liest die Inhalte von sechzehn Speicherstellen aus und stellt sie auf dem Bildschirm dar:

```
10 INPUT "GEBEN SIE DIE 'PEEK'-START-
   ADRESSE EIN";S
20 PRINT
30 FOR L=1 TO 16
40 LET A=PEEK(S)
50 PRINT "SPEICHERSTELLE ";S;"
   ENTHAELT: ";A
60 LET S=S+1
70 NEXT L
80 PRINT "DRUECKE LEERTASTE ZUM
   BETRACHTEN DER NAECHSTEN 16
   SPEICHERSTELLEN"
90 PRINT "ODER RETURN ZUM
   BEENDEN"
100 FOR I=1 TO 1
110 LET C$=INKEY$
120 IF C$ <> CHR$(13) AND C$
   <> " " THEN I=0
130 NEXT I
140 IF C$=CHR$(13) THEN GOTO 160
150 GOTO 30
160 END
```

Die Schleife in den Zeilen 100 bis 130 überprüft die Eingabe über die Tastatur. Wenn das eingegebene Zeichen ein RETURN war (ASCII-Wert 13), wird der Programmlauf beendet. Bei jedem anderen Wert wird der Programmlauf hinter der INPUT-Anweisung neu gestartet.

Wenn Sie es wünschen, kann durch Verwendung der Anweisung PRINT CHR$(A) auch das ASCII-Zeichen der Speicherstelle dargestellt werden. Doch Vorsicht! ASCII-Werte niedriger als dezimal 32 (32 ist der ASCII-Wert für das Leerzeichen) sind nicht einheitlich definiert. Alle ASCII-Werte von 0 bis 31 repräsentieren nicht druckbare Zeichen oder Spezialfunktionen, wie z. B. Cursor-Kontrollzeichen. Die einzige Übereinkunft der verschiedenen Computer-Hersteller ist, daß ASCII 13 normalerweise das RETURN-Zeichen ist und daß ASCII 7 einen Kontrollton produziert (gegebenenfalls vom eingebauten Lautsprecher).

POKE ist die Umkehrfunktion von PEEK. Sie gestattet Ihnen, einen Wert von 0 bis 255 in eine Speicherstelle zu schreiben. Diese Funktion muß mit größter Vorsicht verwendet werden. Schreibt man beispielsweise in einen Speicherbereich, der bereits vom Programm belegt ist, so kann das höchst unerfreuliche Folgen haben. In Maschinensprache geschriebene Routinen können in die entsprechenden Speicherstellen gePOKEd und während des Programmlaufs mit der CALL-Anweisung aktiviert werden.

Viele der neuen Computer gestatten es, bestimmte Positionen auf dem Bildschirm direkt anzusteuern. Doch selbst wenn Ihr Computer über diese Möglichkeit nicht verfügt, läßt sich der Cursor durch eine zusätzliche Routine nach links, rechts, oben und unten über den Bildschirm bewegen. Dazu müssen Sie als erstes wissen, welche ASCII-Werte die Cursor-Kontrolltasten repräsentieren. Das folgende Programm fordert Sie auf, eine Taste zu drükken und gibt dann den entsprechenden ASCII-Wert der gedrückten Taste an:

```
1 REM FINDET DIE ASCII-WERTE FUER
  DIE CURSOR-TASTEN
10 PRINT "DRUECKE EINE TASTE";
20 FOR I=1 TO 1
30 LET K$=INKEY$
40 IF K$="" THEN I=0
50 NEXT I
60 PRINT ASC(K$)
70 GOTO 10
80 END
```

Diese Routine ermöglicht Ihnen zusätzlich, auch die Werte für die RETURN-Taste, ESCape und die Leertaste zu finden. Der Sord-M23-Computer, auf dem die Programme des BASIC-Programmierkurses entwickelt wurden, verwendet den Wert 8 für Cursor links, 28 für Cursor rechts, 29 für Cursor auf- und 30 für Cursor abwärts. Ihr Computer wird wahrscheinlich andere Werte verwenden. Setzen Sie entsprechend die von Ihnen ermittelten Werte in das folgende Programm ein:

```
 10 PRINT CHR$(12): REM VERWENDE
    CLS ODER ENTSPRECHENDEN CODE
 20 FOR L=1 TO 39
 30 PRINT "*";
 40 NEXT L
 50 FOR L=1 TO 22
 60 PRINT CHR$(8);: REM VERWENDE
    CODE FUER 'CURSOR LINKS'
 70 NEXT L
 80 FOR L=1 TO 4
 90 PRINT "@";
100 NEXT L
110 END
```

Es sollte eine Zeile auf dem Bildschirm dargestellt werden, die so aussieht:

**********@@@@**********

Durch die Zeilen 20 bis 40 wird eine Zeile mit 39 Sternen gedruckt. Die Zeilen 50 bis 70 veranlassen den Cursor, um 22 Stellen nach links zu rücken. Durch die Zeilen 80 bis 100 wird dann viermal ein @ gedruckt. Danach ist der Programmlauf beendet. Programmiertechniken wie diese gestatten dem User, den Cursor beliebig über den Bildschirm zu bewegen und Zeichen an neuen Positionen zu drucken.

Um Ihnen zu zeigen, wie man diese Art der Cursor-Kontrolle zur Erstellung einer einfachen Grafik als Ergebnis eines Programms verwenden kann, testen Sie doch einmal das folgende Programm:

```
 10 PRINT "DIESES PROGRAMM ZEICH-
    NET EIN BALKENDIAGRAMM VON
    3 VARIABLEN"
 20 INPUT "GEBEN SIE DREI WERTE EIN ";
    X,Y,Z
 30 PRINT
 40 FOR L=1 TO 2
 50 FOR A=1 TO X
 60 PRINT "*";
 70 NEXT A
 80 PRINT CHR$(13)
 90 NEXT L
100 FOR L=1 TO 2
110 FOR A=1 TO Y
120 PRINT "+";
130 NEXT A
140 PRINT CHR$(13)
150 NEXT L
160 FOR L=1 TO 2
170 FOR A=1 TO Z
180 PRINT "#";
190 NEXT A
200 PRINT CHR$(13)
210 NEXT L
220 PRINT
230 END
```

Das Programm erstellt aus den drei Werten der Variablen ein Balken-Diagramm. Die Balken werden in horizontalen Reihen (von links nach rechts) ausgegeben. Beachten Sie, daß in den Zeilen 80, 140 und 200 jeweils eine PRINT CHR$(13)-Anweisung notwendig ist. Diese wird benötigt, da ein Semikolon am Ende einer PRINT-Anweisung ein RETURN unterdrückt.

Bisher haben wir nur zwei Variablenarten behandelt: numerische und String-Variablen. In BASIC gibt es jedoch mehrere verschiedene Arten von numerischen Variablen, und ein guter Programmierer wird je nach Bedarf einen Typ auswählen (um z. B. Speicherplatz zu sparen oder die Genauigkeit zu erhöhen).

Wenn eine Variable definiert wird, wird eine bestimmte Menge an Speicherplatz automatisch zum Speichern der Variable reserviert. Wird zuvor angegeben, daß die Variable immer integer sein wird, braucht entsprechend weniger Speicherplatz reserviert zu werden. Hat man eine Variable, die eine unbestimmbare Anzahl verschiedener Werte beinhalten wird (z. B. LET GEBIET=PI*RADIUS*RADIUS), muß eine größere Menge an Speicherplatz bereitgestellt werden.

Variablenspezifikation

Während der Entwicklung des Adreßbuch-Programms haben Sie gelernt, String-Variablen zu spezifizieren, indem hinter dem Variablennamen ein $-Zeichen eingegeben wurde (z. B. LET SUSCHL$=MODFLD$(GROSS)). Variablen ohne das „Dollar"-Zeichen sind normale numerische Variablen. Bei der Kennzeichnung der verschiedenen Arten von numerischen Variablen ist der Vorgang jedoch dem bei den String-Variablen ähnlich. Eine Variable mit einem Variablennamen ohne besondere Kennzeichnung wird als reale numerische Variable einfacher Genauigkeit angesehen. Zeichen zur Kennzeichnung sind bei einigen BASIC-Versionen: % zur Kennzeichnung von Integer-Variablen, ! zur Kennzeichnung einfach genauer Variablen sowie # zur Spezifizierung einer doppelt genauen Variablen (die Variable kann dann doppelt soviele Nachkommastellen speichern). Betrachten Sie das folgende Programmbeispiel, in dem die eben angesprochenen Zeichen verwendet werden:

BASIC-Dialekte

INKEY$

Beim Lynx müssen Sie im ersten Programm die Zeilen 110, 120 und 140 wie folgt ersetzen:

```
110 C=KEYN
120 IF (C <> 13)
    AND (C <>
32)
    THEN I=0
140 IF C=13 THEN
    GOTO 160
```

CURSOR MOVEMENT

Das dritte Beispielprogramm läuft zwar, doch erhalten Sie bei einigen Computern, wie etwa Dragon und Acorn B, nicht das gewünschte Ergebnis.

PEEK

Beim Acorn B müssen Sie PEEK(S) durch ?S ersetzen.

```
70 LET SPIELER$="DIRK": REM EINE
   STRING-VARIABLE
80 LET PUNKTZAHL%=0: REM EINE
   INTEGER-VARIABLE
90 LET PI!=3.1416: REM EINE EINFACH
   GENAUE VARIABLE
100 LET BEREICH#=PI*R*R: REM
   DOPPELT GENAUE VARIABLE
110 LET RUNDEN=6: REM ANGENOM-
   MEN ALS EINFACH GENAUE
   VARIABLE
```

Wir müssen jedoch anmerken, daß nicht alle BASIC-Dialekte über diese drei Variablen-Typen verfügen. Der Spectrum beispielsweise bietet keine Integer-Variablen. Integer-Werte werden als einfache präzise reale Zahlen gespeichert. Außerdem verfügt der Spectrum nicht über doppelt präzise Zahlen. Dafür werden beim Spectrum-BASIC einfach genaue Zahlen auf neun Stellen genau berechnet.

Computer, die über Integer-Variablen verfügen, verwenden normalerweise zwei Bytes zum Speichern der Zahl, wobei die Zahl in einem Bereich von -32768 bis 32767 liegen kann. Dieser Bereich reicht im allgemeinen für Variablen wie Punktzahlen, Spielerzahlen oder Zähler von FOR. . .NEXT-Schleifen aus. Da zum Speichern der Zahl nur zwei Bytes verwendet werden, spart die Verwendung von Integer-Variablen Speicherplatz.

Dies ist die vollständige Spectrum-Fassung des Adreßbuch-Programms. Weitere Unterschiede zwischen den verschiedenen BASIC-Varianten werden in der nächsten Folge dieses Kurses veröffentlicht. Dabei wird auf dieses Listing Bezug genommen.

```
 1 REM *ENTWICKLE DATEN—DATEI*
 2 DIM N$(50,30)
 3 LET N$(1)="@ ERST"
 4 INPUT "DATENCASSETTE EINLEGEN, PLAY DRUECKEN,
   DANACH 'ENTER'";A$
 5 SAVE "NFLD" DATA N$( )
 6 INPUT "BAND ZURUECKSPULEN, PLAY DRUECKEN,
   DANACH 'ENTER'"; A$
 7 VERIFY "NFLD" DATA N$( )
 8 STOP
```

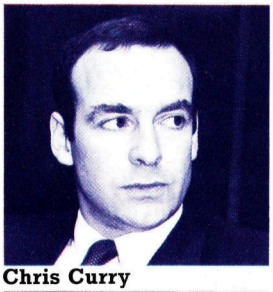
Dies ist das Initialisierungsprogramm, das auf der Cassette den Bereich definiert, der für den ersten Programmlauf notwendig ist. Danach spulen Sie die Daten-Cassette zurück, laden das Hauptprogramm und starten es mit RUN.

```
 10 REM 'HAUPTPROGRAMM'
 20 REM * INITIL *
 30 GOSUB 1000
 40 REM * BGRUES *
 50 GOSUB 3000
 60 FOR M=1 TO 1
 70 LET M=0
 80 REM * AUWAHL *
 90 GOSUB 3500
100 REM * AUSFUH *
110 GOSUB 4000
120 IF WAHL=9 THEN LET M=1
130 NEXT M
140 STOP

1000 REM * INITIL * UNTERROUTINE
1010 GOSUB 1100
1020 GOSUB 1400
1030 GOSUB 1600
1090 RETURN

1100 REM *CREARR* UNTERROUTINE
1110 DIM N$(50,30)
1120 DIM M$(50,30)
1130 DIM S$(50,30)
1140 DIM T$(50,15)
1150 DIM C$(50,15)
1160 DIM R$(50,15)
1170 DIM X$(50,30)
1180 DIM B$(30):DIM Z$(30)
1190 DIM U$(30):DIM W$(15)
1210 LET GROSS=0
1220 LET VMOD=0
1230 LET SRTD=1
1240 LET CURR=0
1250 LET Z$="@ERST"
1260 LET Q$=B$
1300 RETURN

1400 REM *LSINDT* UNTERROUTINE
1405 INPUT "DATENCASSETTE EINLEGEN, PLAY DRUECKEN, &
     'ENTER' DRUECKEN";A$
1410 LOAD "NFLD" DATA N$( )
1420 IF N$(1)=Z$ THEN LET Q$=Z$:RETURN
1430 LOAD "MFLD" DATA M$( )
1440 LOAD "SFLD" DATA S$( )
1450 LOAD "TFLD" DATA T$( )
1460 LOAD "CFLD" DATA C$( )
1470 LOAD "TELFLD" DATA R$( )
1480 LOAD "NDXFLD" DATA X$( )
1485 INPUT "STOPPEN SIE DEN RECORDER, & DRUECKEN SIE
     'ENTER'";A$
1490 REM 'DATEIGROESSE'
1500 LET GROSS=51
1510 FOR L=1 TO 50
1520 IF N$(L)=B$ THEN LET GROSS=L:LET L=50
1530 NEXT L
1540 RETURN

1600 REM *SETFLG* UNTERROUTINE
1640 IF Q$=Z$ THEN LET GROSS=1
1690 RETURN

3000 REM * BGRUES * UNTERROUTINE
3010 CLS
3020 PRINT:PRINT:PRINT:PRINT
3060 PRINT TAB(13);"*WILLKOMMEN ZUM*"
3070 PRINT TAB(11);"*ADRESSBUCH-PROGRAMM*"
3080 PRINT TAB(8);"*DES COMPUTERKURSES*"
3090 PRINT
3100 PRINT TAB(1);"(DRUECKE DIE LEERTASTE, UM
     FORTZUFAHREN)"
3110 FOR L=1 TO 1
3120 IF INKEY$ <> " " THEN L=0
3130 NEXT L
3140 CLS
3150 RETURN

3500 REM * AUWAHL * UNTERROUTINE
3520 IF Q$=Z$ THEN GOSUB 3860:RETURN
3540 REM 'CHMENU'
3550 CLS
3560 PRINT "WOLLEN SIE"
3570 PRINT:PRINT:PRINT
3600 PRINT "1. VERZEICHNIS DURCH NAMEN SUCHEN"
3610 PRINT "2. NAMEN DURCH TEIL EINES NAMENS SUCHEN"
3620 PRINT "3. VERZEICHNISSE NACH STADTANGABEN SUCHEN"
3630 PRINT "4. VERZEICHNISLISTE DURCH INITIALEN"
3640 PRINT "5. LISTE ALLER VERZEICHNISSE"
3650 PRINT "6. HINZUFUEGEN EINER ADRESSE"
3660 PRINT "7. AENDERN EINER ADRESSE"
3670 PRINT "8. LOESCHEN EINER ADRESSE"
3680 PRINT "9. PROGRAMM BEENDEN UND DATEN SPEICHERN"
3690 PRINT:PRINT
3710 REM 'INWAHL'
3750 PRINT "WAEHLEN SIE (1—9)"
3760 FOR L=1 TO 1
3770 FOR I=1 TO 1
3780 LET A$=INKEY$
3790 IF A$="" THEN I=0
3800 NEXT I
3810 LET WAHL=CODE A$-48
3820 IF (WAHL < 1) OR (WAHL > 9) THEN LET L=0
3840 NEXT L
3850 RETURN

3860 REM *ERSTLA* UNTERROUTINE
3870 LET WAHL=6
3880 CLS
3890 PRINT
3900 PRINT TAB(2);"ES SIND KEINE VERZEICHNISSE"
3910 PRINT TAB(2);"IN DER DATEI! SIE MUESSEN"
3920 PRINT TAB (2);"MIT DER EINGABE EINES VERZEICHNISSES
     BEGINNEN"
3930 PRINT
3940 REM *FORTFAHREN*
3950 GOSUB 3100
3990 RETURN

4000 REM *AUSFUH* UNTERROUTINE
4040 IF WAHL = 1 THEN GOSUB 5700
4050 REM 2 IST *FNDNMS*
4060 REM 3 IST *FNDSTD*
4070 REM 4 IST *FNDINT*
4080 REM 5 IST *LSTVER*
4090 IF WAHL = 6 THEN GOSUB 4200
4100 IF WAHL = 7 THEN GOSUB 6600
4110 IF WAHL = 8 THEN GOSUB 7500
4120 IF WAHL = 9 THEN GOSUB 5000
4140 RETURN

4200 REM *ADDVER* UNTERROUTINE
4210 CLS
4220 INPUT "GIB NAMEN EIN";N$(GROSS)
4230 INPUT "GIB STRASSE EIN";S$(GROSS)
4240 INPUT "GIB STADT EIN";T$(GROSS)
4250 INPUT "GIB STAAT EIN";C$(GROSS)
4260 INPUT "GIB TELEFONNUMMER EIN";R$(GROSS)
4270 LET VMOD=1:LET SRTD=0
4280 LET X$(GROSS)=STR$(GROSS)
4290 LET Q$=""
4300 GOSUB 4500
4310 LET WAHL=0
4320 LET GROSS=GROSS+1
4350 RETURN

4500 REM *MODNAME* UNTERROUTINE
4510 REM UMWANDLUNG IN GROSSBUCHSTABEN
4520 LET D$=N$(GROSS):LET P$=""
4530 FOR L=1 TO LEN(D$)
4540 LET A$=D$(L)
4550 LET T=CODE A$
4560 IF T > =97 THEN LET T=T-32
4570 LET A$=CHR$ T
```

```
4580 LET P$=P$+A$
4590 NEXT L
4600 LET D$=P$:LET P$="":LET A$="":LET T=LEN(D$):LET S=0
4610 REM LOKALISIERT ERSTE LEERSTELLE
4620 FOR L=1 TO T
4630 IF D$(L)=" " THEN LET S=L:LET L=T
4640 NEXT L
4650 REM ENTFERNT UNNOETIGE ZEICHEN UND SPEICHERT
     VORNAMEN IN P$
4660 FOR L=1 TO S-1
4670 IF CODE(D$(L)) > 64 THEN LET P$=P$+D$(L)
4680 NEXT L
4690 REM ENTFERNT UNNOETIGE ZEICHEN UND SPEICHERT
     FAMILIENNAMEN IN A$
4700 FOR L=S+1 TO LEN(D$)
4710 IF CODE(D$(L)) > 64 THEN LET A$=A$+D$(L)
4720 NEXT L
4730 LET M$(GROSS)=A$+" "+P$
4740 LET P$="":LET A$="":LET S=0
4750 RETURN

5000 REM *ENPROG* UNTERROUTINE
5010 IF (VMOD=0) AND (SRTD=1) THEN RETURN
5020 IF (VMOD=1) AND (SRTD=0) THEN GOSUB 5200
5030 GOSUB 5600
5040 RETURN

5200 REM *SRTVER* UNTERROUTINE
5210 FOR K=1 TO 1
5220 LET S=0
5230 FOR L=1 TO GROSS-2
5240 LET T=L+1
5250 IF M$(L) > M$(T) THEN GOSUB 5400
5260 NEXT L
5270 IF S=1 THEN LET K=0
5280 NEXT K
5290 LET SRTD=1
5300 RETURN

5400 REM *VTAVER* UNTERROUTINE
5410 LET U$=N$(L):LET N$(L)=N$(T):LET N$(T)=U$
5420 LET U$=M$(L):LET M$(L)=M$(T):LET M$(T)=U$
5430 LET U$=S$(L):LET S$(L)=S$(T):LET S$(T)=U$
5440 LET W$=T$(L):LET T$(L)=T$(T):LET T$(T)=W$
5450 LET W$=C$(L):LET C$(L)=C$(T):LET C$(T)=W$
5460 LET W$=R$(L):LET R$(L)=R$(T):LET R$(T)=W$
5470 LET X$(L)=STR$(L)
5480 LET X$(T)=STR$(T)
5490 LET S=1
5500 RETURN

5600 REM *SPEVER* UNTERROUTINE
5605 INPUT "LEGE SPEICHERCASSETTE EIN & DRUECKE 'ENTER'";
     A$
5610 SAVE "NFLD" DATA N$( )
5620 SAVE "MFLD" DATA M$( )
5630 SAVE "SFLD" DATA S$( )
5640 SAVE "TFLD" DATA T$( )
5650 SAVE "CFLD" DATA C$( )
5660 SAVE "TELFLD" DATA R$( )
5670 SAVE "INDFLD" DATA X$( )
5680 INPUT "STOPPE DIE CASSETTE & DRUECKE 'ENTER'";A$
5690 RETURN

5700 REM *FNDVER* UNTERROUTINE
5710 CLS
5720 IF SRTD=0 THEN GOSUB 5200
5730 PRINT:PRINT
5740 PRINT TAB(9);"SUCHEN EINES VERZEICHNISSES"
5750 PRINT TAB(16);"NACH NAMEN"
5760 PRINT
5770 PRINT TAB(9);"GEBEN SIE DEN KOMPLETTEN NAMEN EIN"
5780 PRINT TAB(7);"(VORNAMEN, FAMILIENNAME)"
5790 PRINT:PRINT
5800 INPUT "DER NAME LAUTET ";N$(GROSS)
5810 GOSUB 4500
5820 LET U$=M$(GROSS)
5830 LET BTM=1
5840 LET TP=GROSS-1
5850 FOR X=1 TO 1
5860 LET MD=INT((BTM+TP)/2)
5870 IF M$(MD) <> U$ THEN LET X=0
5880 IF M$(MD) < U$ THEN LET BTM=MD+1
5890 IF M$(MD) > U$ THEN LET TP=MD-1
5900 IF BTM > TP THEN LET X=1
5910 NEXT X
5920 IF BTM > TP THEN LET CURR=0
5930 IF BTM < = TP THEN LET CURR=MD
5940 IF CURR=0 THEN GOSUB 6400:RETURN
5950 CLS
5960 PRINT
5970 PRINT TAB(9);"*VERZEICHNIS GEFUNDEN*"
5980 PRINT
5990 PRINT "NAME: ",N$(CURR)
6000 PRINT "STRASSE: ",S$(CURR)
6010 PRINT "STADT: ",T$(CURR)
6020 PRINT "STAAT: ",C$(CURR)
6030 PRINT "TELEFON: ",R$(CURR)
6040 PRINT
6050 PRINT TAB(3);"DRUECKE TASTE ZUM AUSDRUCKEN"
6060 PRINT TAB(3);"ODER LEERTASTE, UM FORTZUFAHREN"
6070 FOR I=1 TO 1
6080 LET A$=INKEY$
6090 IF A$="" THEN LET I=0
6100 NEXT I
6110 IF A$ <> " " THEN GOSUB 6200
6120 RETURN

6200 REM *LSTCUR* UNTERROUTINE
6210 LPRINT
6220 LPRINT "NAME: ",N$(CURR)
6230 LPRINT "STRASSE: ",S$(CURR)
6240 LPRINT "STADT: ",T$(CURR)
6250 LPRINT "STAAT: ",C$(CURR)
6260 LPRINT "TELEFON: ",R$(CURR)
6270 LPRINT:LPRINT
6280 RETURN

6400 REM *NOTVER* UNTERROUTINE
6410 CLS
6420 PRINT TAB(7);"*VERZEICHNIS NICHT GEFUNDEN*"
6430 PRINT TAB(4);"*IN DER FORM: ";N$(GROSS);" *"
6440 PRINT
6450 REM 'FORTFAHREN'
6460 GOSUB 3100
6470 RETURN

6600 REM *MODVER* UNTERROUTINE
6610 CLS
6620 PRINT:PRINT:PRINT
6630 LET E$=CHR$ 13
6640 PRINT TAB(3);"*UM EIN VERZEICHNIS ZU MODIFIZIEREN*"
6650 PRINT TAB(3);"*LOKALISIERE ES ZUERST*"
6660 GOSUB 5720
6670 IF CURR=0 THEN RETURN
6680 PRINT
6690 PRINT TAB(3);"MODIFIZIERE NAMEN?"
6700 PRINT
6710 PRINT TAB(3);"DRUECKE 'ENTER' FUER NEUEN NAMEN"
6720 PRINT TAB(3);"ODER LEERTASTE FUER NAECHSTES FELD"
6730 FOR I=1 TO 1
6740 LET A$=INKEY$
6750 IF (A$ <> E$) AND (A$ <> " ") THEN LET I=0
6760 NEXT I
6770 IF A$=E$ THEN INPUT "NEUER NAME"; N$(CURR)
6780 IF A$=E$ THEN VMOD=1
6790 IF A$=E$ THEN SRTD=0
6800 IF A$=E$ THEN LET N$(GROSS)=N$(CURR)
6810 IF A$=E$ THEN GOSUB 4500
6820 IF A$=E$ THEN LET M$(CURR)=M$(GROSS)
6830 PRINT
6840 PRINT TAB(3);"MODIFIZIERE STRASSE?"
6850 PRINT
6860 PRINT TAB(3);"DRUECKE 'ENTER' FUER NEUE STRASSE"
6870 PRINT TAB(3);"ODER LEERTASTE FUER NAECHSTES FELD"
6880 FOR I=1 TO 1
6890 LET A$=INKEY$
6900 IF (A$ <> E$) AND (A$ <> " ") THEN LET I=0
6910 NEXT I
6920 IF A$=E$ THEN LET VMOD=1
6930 IF A$=E$ THEN INPUT "NEUE STRASSE";S$(CURR)
6940 PRINT
6950 PRINT TAB(3);"MODIFIZIERE STADT?"
6960 PRINT
6970 PRINT TAB(3);"DRUECKE 'ENTER' FUER NEUE STADT"
6980 PRINT TAB(3);"ODER LEERTASTE FUER NAECHSTES FELD"
6990 FOR I=1 TO 1
7010 LET A$=INKEY$
7020 IF (A$ <> E$) AND (A$ <> " ") THEN LET I=0
7030 NEXT I
7040 IF A$=E$ THEN LET VMOD=1
7050 IF A$=E$ THEN INPUT "NEUE STADT";T$(CURR)
7060 PRINT
7070 PRINT TAB(3);"MODIFIZIERE STAAT?"
7080 PRINT
7090 PRINT TAB(2);"DRUECKE 'ENTER' FUER NEUEN STAAT"
7100 PRINT TAB (2);"ODER LEERTASTE FUER NAECHSTES FELD"
7110 FOR I=1 TO 1
7120 LET A$=INKEY$
7130 IF (A$ <> E$) AND (A$ <> " ") THEN LET I=0
7140 NEXT I
7150 IF A$=E$ THEN LET VMOD=1
7160 IF A$=E$ THEN INPUT "NEUER STAAT";C$(CURR)
7170 PRINT
7180 PRINT TAB(3);"MODIFIZIERE TELEFONNUMMER?"
7190 PRINT
7200 PRINT "DRUECKE 'ENTER' FUER NEUE TELEFONNUMMER"
7210 PRINT "ODER LEERTASTE, UM FORTZUFAHREN"
7220 FOR I=1 TO 1
7230 LET A$=INKEY$
7240 IF (A$ <> E$) AND (A$ <> " ") THEN LET I=0
7250 NEXT I
7260 IF A$=E$ THEN LET VMOD=1
7270 IF A$=E$ THEN INPUT "NEUE NUMMER";R$(CURR)
7280 RETURN

7500 REM *LOEVER* UNTERROUTINE
7510 CLS
7520 PRINT:PRINT:PRINT:PRINT
7530 LET E$=CHR$ 13
7540 PRINT TAB (3);"*UM EIN VERZEICHNIS ZU LOESCHEN*"
7550 PRINT TAB(3);"*LOKALISIERE ES ZUERST*"
7560 GOSUB 5720
7570 IF CURR=0 THEN RETURN
7580 PRINT
7590 PRINT TAB(3);"VERZEICHNIS LOESCHEN?"
7600 PRINT TAB(3);"WARNUNG — — DIES IST NICHT
     WIDERRUFBAR"
7610 PRINT
7620 PRINT TAB(5);"DRUECKE 'ENTER' ZUM LOESCHEN"
7630 PRINT TAB(5);"ODER LEERTASTE, UM FORTZUFAHREN"
7640 FOR I=1 TO 1
7650 LET A$=INKEY$
7660 IF (A$ <> E$) AND (A$ <>" ") THEN LET I=0
7670 NEXT I
7680 IF A$=" " THEN RETURN
7690 FOR L=CURR TO GROSS-2
7700 LET T=L+1
7710 LET N$(L)=N$(T)
7720 LET M$(L)=M$(T)
7730 LET S$(L)=S$(T)
7740 LET T$(L)=T$(T)
7750 LET C$(L)=C$(T)
7760 LET R$(L)=R$(T)
7770 LET X$(L)=X$(T)
7780 NEXT L
7790 LET VMOD=1
7800 LET GROSS=GROSS-1
7810 RETURN
```

Harter Gegner

Die Firma Acorn Computer dokumentiert mit ihren leistungsfähigen Computern den hohen Stand der britischen Technologie.



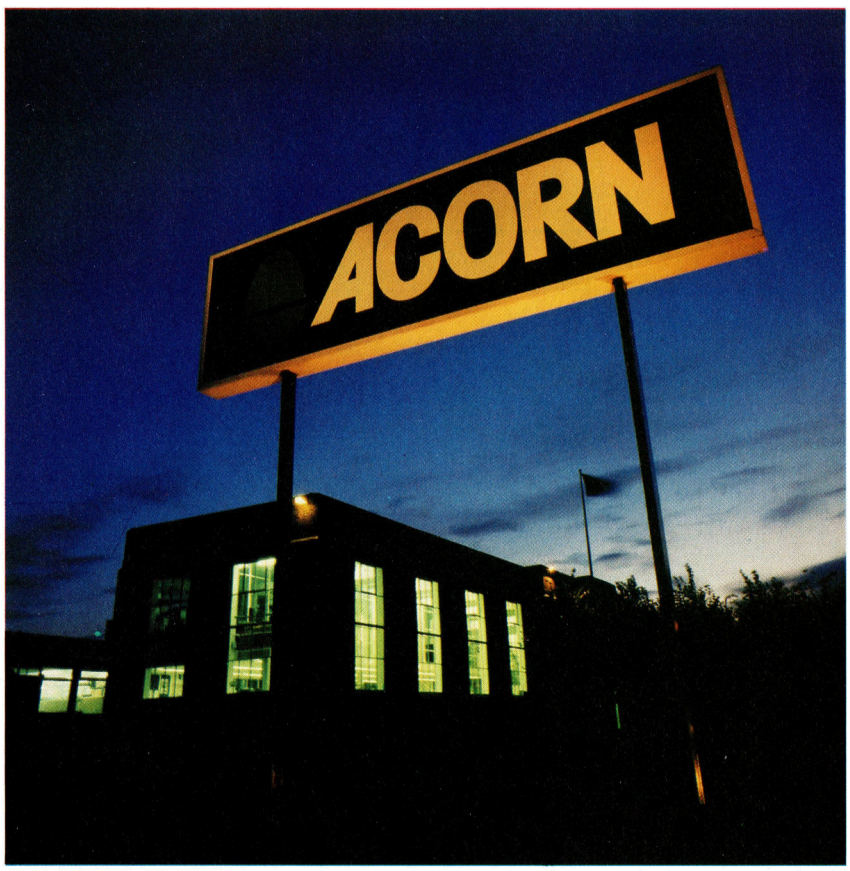
Chris Curry

Herman Hauser

Der Gründer von Acorn, Chris Curry, war ein ehemaliger Angestellter und Freund von Sir Clive Sinclair. Curry kam 1965 zu Sinclair Radionics, nachdem Sinclair ihm für elf Pfund die Woche eine Stelle als Entwicklungsingenieur angeboten hatte.

Bei Sinclair Radionics war Curry für die Projektgruppe zuständig, die 1971 den Taschenrechner Executive fertigstellte. In den folgenden fünf Jahren beschäftigte er sich mit der Entwicklung jener Rechner, die heute als Vorläufer der Heimcomputer betrachtet werden. 1975 reduzierte Sinclair Radionics seine Verkaufsaktivitäten, und Curry arbeitete für Sinclair auf freier Basis weiter. Diese Unternehmung lief unter dem Namen „Science of Cambridge" und befaßte sich mit dem Vertrieb elektronischer Komponenten in Bausatzform.

Als gut verkäuflich erwies sich eine Armbanduhr mit integriertem Rechner. Die Einplatinencomputer, die zu dieser Zeit in den USA auf dem Vormarsch waren, inspirierten Curry. Und er begann mit einer Eigenentwicklung. Dieser Bausatz kam unter der Bezeichnung MK 14 (Microprocessor Kit [= Bausatz] mit 14 Chips) auf den Markt. Im Lieferumfang enthalten war ein National Semiconductor Microprozessor, dessen Leistungsfähigkeit bei 256 Byte RAM lag. Dazu stand separater Speicherplatz für den Bildaufbau zur Verfügung. Die für den Betrieb einer achtstelligen LED-Anzeige erforderlichen Komponenten rundeten das Lieferpaket ab.



Wortspiel als Name

Curry fiel auf, daß das Unternehmen stets telefonisch um Rat gefragt wurde. Um die anstehenden Fragen beantworten zu können, stellte er Herman Hauser, einen Absolventen der Universität Cambridge, an. Bald jedoch zeigte sich, daß Currys und Sinclairs Zielsetzungen in verschiedene Richtungen gingen. Curry schloß daraus, daß es an der Zeit sei, eine eigene Gesellschaft zu gründen. Er realisierte dies mit Hauser als neuem Partner und nannte die Gesellschaft Cambridge Processor Unit. Reduziert man dies auf die Anfangsbuchstaben CPU, wird so das Wortspiel deutlich.

Der Erfolg des MK 14 und die Entwicklungen in den USA hatten bewiesen, daß der Konsument nach einem kompletten System mit integriertem BASIC suchte. CPU entwickelte daraufhin eine sehr schnelle BASIC-Version, und man beschloß, dieses BASIC in einen Rechner zu integrieren und in der Form auf den Markt zu bringen. Der Rechner erhielt den Namen Atom, und zur Vermarktung bzw. zum Vertrieb wählte man den Firmennamen Acorn. Das Folgeprodukt, hauptsächlich für die Verwendung in Laboratorien und an Universitäten entwikkelt, ging unter der Bezeichnung Proton in die Entwicklung.

1981, Proton war noch im Stadium der Vorproduktion, erfuhr Curry, daß die BBC nach einem Rechner suchte. Dieser sollte eine Fernsehserie begleiten, die Computerwissen vermittelte. Curry reagierte darauf, indem er die Möglichkeiten des 6502-Prozessors demonstrierte – allerdings nicht mit dem Proton, sondern mit einem eigens entwickelten System.

Die BBC hatte vorgegeben, daß der Rechner für Einsteiger leicht bedienbar und zugleich erweiterungsfähig sein müsse. Zudem sollte das Preis-Leistungs-Verhältnis stimmen (Vorgabe war ein Endverbraucherpreis von maximal 200 Pfund). Trotz harten Wettbewerbs durch Sinclair Research bekam Acorn den Auftrag und entwickelte den Acorn B, der später in hohen Stückzahlen produziert wurde.

Commodore 16

Der Commodore 16 ist der preisgünstigere der beiden Computermodelle, die Commodore 1984 auf den Markt brachte. Die zweite Maschine trägt die Bezeichnung Plus/4, sie verfügt über einen größeren Arbeitsspeicher und integrierte Software.

Der Commodore 16 stellt das Nachfolgemodell des VC 20 dar. Der Plus/4 wie auch der C 16 verfügen über ein leistungsfähiges BASIC, das zusätzliche Diskettenbefehle, eine Reihe von Hilfsprogrammen und Befehle für Tonsteuerung und Grafik beinhaltet. Befehle wie DO... WHILE und LOOP... UNTIL... EXIT ermöglichen zudem eine strukturierte Programmierung.

Wenn sich auch das Gehäuse und die Tastatur des Plus/4 radikal von allem unterscheiden, was Commodore bisher auf den Markt gebracht hat, so erinnert der C 16 aufgrund seiner äußeren Gestaltung doch stark an den VC 20. Die Gehäusefarbe ist ein tiefes Schwarz, und die Anordnung der hellgrauen Tasten unterscheidet sich von den früheren Modellen. Die Tasten des 16 sind wie bei seinem Vorgänger groß und übersichtlich angeordnet. Eine interessante Neuheit ist die HELP-Taste (eigentlich die Funktionstaste F8), mit der bei einem Programmabbruch aufgrund eines Syntax-Fehlers die entsprechende Programmzeile dargestellt und der fehlerhafte Teil blinkend angezeigt werden kann. Enthält eine Zeile mehr als einen Befehl, blinkt allerdings die gesamte Zeile von der Position des Fehlers an.

Seltsamerweise hat Commodore mit diesen beiden Maschinen Hardware hergestellt, die nicht – wie frühere Geräte – „aufwärtskompatibel" ist: Auf keinem der beiden neuen Geräte läuft die Software des VC 20 und des Commodore 64. Selbst die Buchsen für die Joysticks haben ein Format, das von dem als Standard angesehenen neunpoligen D-Stecker abweicht, und auch der Cassettenrecorderanschluß wurde umgebaut. Die Schnittstellen für das Diskettenlaufwerk und den Monitor blieben unverändert.

Keine Spriteprogramme

Eine wesentliche Verschlechterung – hauptsächlich für Programmierer – ist das Fehlen von Sprites. Zwar gab es auf dem VC 20 diese Möglichkeit noch nicht, durch den Commodore 64 (und vergleichbare Maschinen) aber hatte sich der Käufer daran gewöhnt, mit der Spritetechnik ausgefeilte Bewegungsgrafiken entwickeln zu können.

Beim Einschalten des Gerätes zeigt der Bildschirm das gewohnte Auftaktbild mit der Information, daß 12277 Bytes frei zur Verfügung stehen. Das integrierte BASIC 3.5 ist die fünfte BASIC-Version, die für Commodoregeräte entwickelt wurde. Die erste Version 1.0 enthielt keine Befehle für den Diskettenzugriff, während Version 2 (auf dem VC 20 und dem Commodore 64 eingesetzt) diese Funktionen nur sehr umständlich steuern konnte. März 1980 wurde Version 2 von der verbesserten Version 4 abgelöst, die die 80-Zeichen-Darstellung des Pet unterstützte und auch auf dem 8032, 8096 und 8296 eingesetzt werden konnte.

Version 3.5 gleicht Version 4, besitzt aber eine Reihe zusätzlicher Befehle. Im Vergleich mit dem VC 20 verfügt die Version 3.5 über mehr als 50 neue Befehle und Funktionen, darunter Hilfsroutinen für Programmentwicklung und Fehlersuche, Möglichkeiten der strukturierten Programmierung und Befehle für Tonsteuerung und Grafik.

Der Befehl MONITOR ruft den Monitor auf. Dabei können mit einfachen Befehlen wie „C" zwei Speicherbereiche miteinander verglichen und deren Unterschiede angezeigt werden, während „S" den bearbeiteten Block auf Diskette oder Band speichert.

Nur wenig verändert wurden die Maschinencoderoutinen für die Ein- und Ausgabe. Die IOINIT Funktion des 64, mit der Ein- und

Gedacht als Nachfolger des VC 20, verfügt die neue Maschine über einen Speicher von 16 K und ein verbessertes BASIC. Der Commodore 16 ist in seinen Schnittstellen und der Software mit dem Plus/4 kompatibel, nicht aber mit früheren Geräten der Commodorereihe wie dem VC 20 oder dem beliebten Commodore 64.

Ausgabegeräte und Programmcartridges angesteuert wurden, ist dabei an der Speicherstelle $FF81 verfügbar.

Die hochauflösende Grafik wird mit der Technik des „Bit Mapping" gesteuert und verfügt über 320x160 Pixel, wobei eine Mehrfarbendarstellung mit 160x160 Bildpunkten möglich ist. Der Befehl GRAPHIC und auch die Unterteilung des Bildschirms sind weitaus einfacher zu handhaben als die PEEKs und POKEs des 64. Der geteilte Bildschirm verfügt jedoch nur über fünf Textzeilen am unteren Rand des Bildschirms. Mit dem CHAR-Befehl läßt sich Text an jede beliebige Stelle des Bildschirms setzen:

```
CHAR 1,0,0, "DIES IST DIE ERSTE ZEILE"
```

schreibt den Text an den oberen Bildschirmrand, wobei 1 die Farbe der Zeichen angibt und die beiden Nullen die Spalte und Zeile festlegen. Die Zahl 1 läßt den Text in Negativdarstellung erscheinen, während 0 diesen Effekt abschaltet. Tritt ein Programmierfehler auf, wird der Bildschirm wieder auf GRAPHIC 0 geschaltet, die „reine" Textdarstellung.

Der Befehl DRAW erinnert an das LOGO-ähnliche DRAW der MSX-Maschinen. Ein Quadrat wird dabei folgendermaßen gezeichnet:

```
DRAW, 10,10 TO 10,60 TO 60,60 TO 60,10
TO 10,10
```

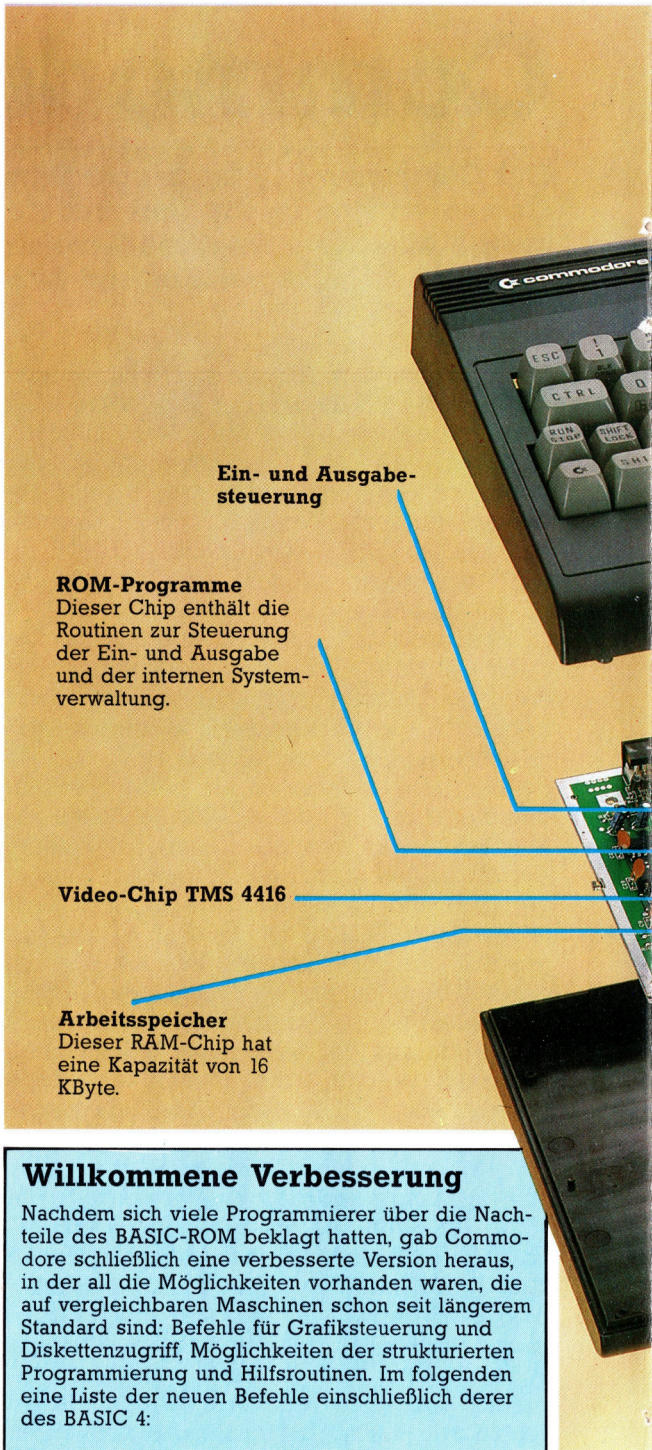
In unserem Beispiel ist der erste Parameter nicht definiert. Daher bestimmt die zuletzt angegebene Farbe die des Quadrates, – ein individueller Farbwert läßt sich natürlich ebenfalls angeben. Auch der Befehl BOX zeichnet Rechtecke auf den Schirm. Dabei werden die vier Ecken festgelegt, und über einen „Farb"-Parameter wird die Farbe angegeben, die das Rechteck „ausfüllen" soll.

Mit dem Befehl CIRCLE lassen sich außer Achtecken, Diamanten und Dreiecken auch „normale" Kreise sowie Ellipsen zeichnen. Die eckigen Formen werden über die Angabe bestimmter Winkel auf den Schirm gebracht. PAINT füllt die entstandene Figur mit der Farbe des Umrisses oder mit einer der Vordergrundfarben. Die Formen können mit SSHAPE (SAVE SHAPE) auf Diskette gespeichert und mit GSHAPE (GET SHAPE) wieder eingelesen werden.

Vom BASIC aus lassen sich 16 Farben dem Hintergrund, Vordergrund, Multicolour 1, Multicolour 2 und dem Rand zuordnen, wobei die Intensität der Farben mit Werten zwischen 0 und 7 bestimmt werden kann. Der Befehl DRAW kann nur Farbwerte verarbeiten, die in einem der fünf Bereiche bereits eingesetzt sind.

Nach der Klangpalette, die der SID-(Sound Interface Device) Chip des 64 möglich macht, bieten die zwei Tonkanäle des Commodore 16 nur ein schwaches Bild. Vergleichbare Maschinen dieser Generation verwenden den „General Instruments"-Chip mit drei Tonkanälen und einem Kanal für Rauschen.

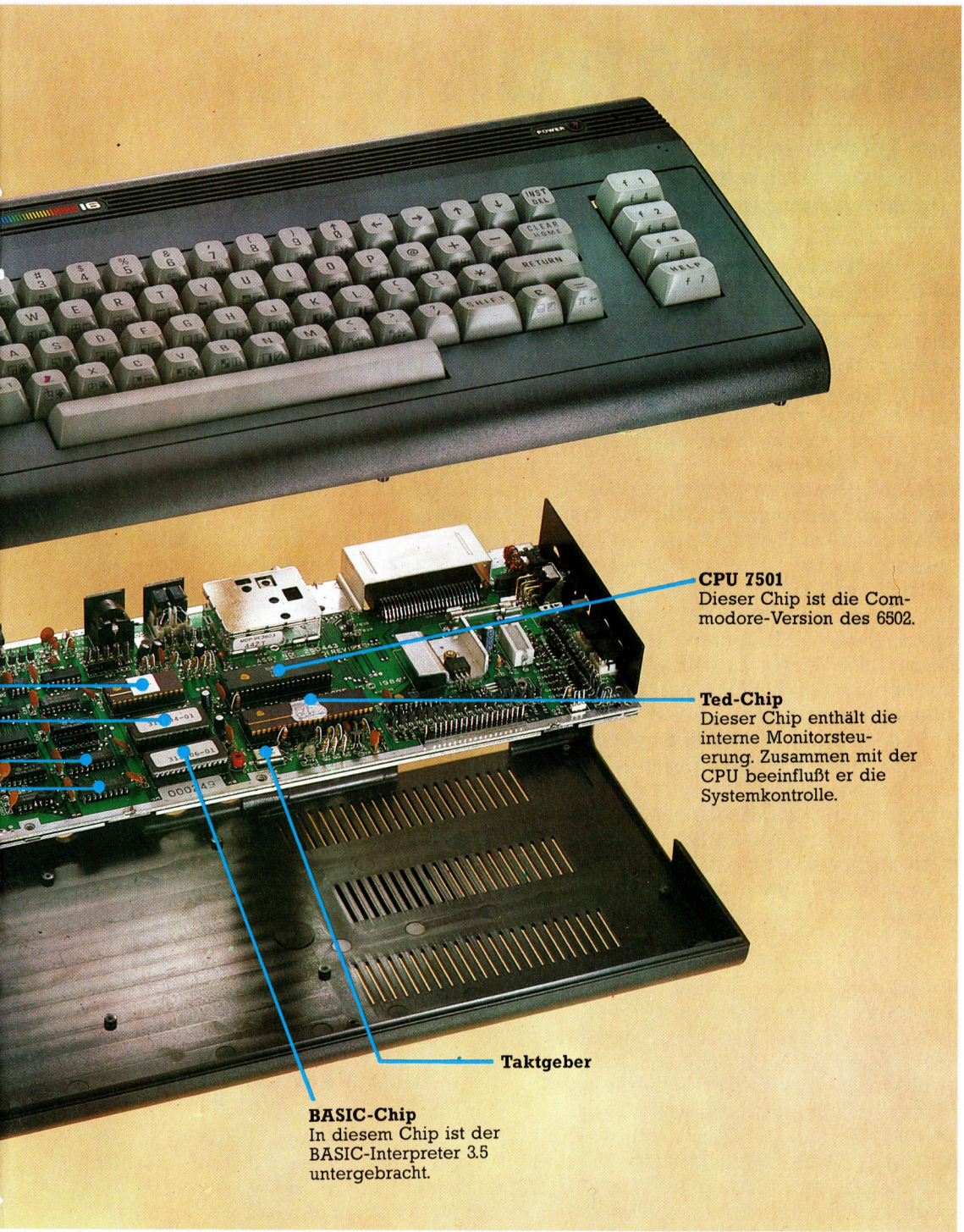
Die Tonsteuerungsbefehle brauchen jedoch nicht mehr wie beim C 64 mit POKE an die Speicherstellen 54272 bis 54296 gesetzt zu



Willkommene Verbesserung

Nachdem sich viele Programmierer über die Nachteile des BASIC-ROM beklagt hatten, gab Commodore schließlich eine verbesserte Version heraus, in der all die Möglichkeiten vorhanden waren, die auf vergleichbaren Maschinen schon seit längerem Standard sind: Befehle für Grafiksteuerung und Diskettenzugriff, Möglichkeiten der strukturierten Programmierung und Hilfsroutinen. Im folgenden eine Liste der neuen Befehle einschließlich derer des BASIC 4:

Funktionen	Hilfsroutinen	Struktur	Grafik	BASIC 4
RGR	AUTO	DO	GRAPHIC	DIRECTORY
RGCL	TRON	WHILE	SCNCLR	DSAVE
RLUM	TROFF	LOOP	PAINT	DLOAD
JOY	HELP	UNTIL	CHAR	HEADER
RDOT	MONITOR	EXT	BOX	SCRATCH
INSTR	DELETE	ELSE	CIRCLE	COLLECT
DEC	RENUMBER	PUDEF	GSHAPE	COPY
HEX$	KEY	USING	SSHAPE	RENAME
SOUND	ERR$		DRAW	BACKUP
VOL	TRAP		LOCATE	
	RESUME		COLOR	



Commodore 16

PREIS
400 Mark

ABMESSUNGEN
76 x 203 x 406 mm

ZENTRALEINHEIT
MOS 7501, 0,89 bis 1,76 MHz

SPEICHERKAPAZITÄT
16 K RAM (davon 12 K für den Anwender verfügbar), 32 K ROM einschließlich Betriebssystem und BASIC-Interpreter.

BILDSCHIRMDARSTELLUNG
Text: 25 Zeilen mit je 40 Zeichen, Grafik: 320 x 160 Pixel. Fünf Grafikmodi: Text, Hochauflösung, Hochauflösung mit 5 Zeilen Text, Mehrfarbendarstellung, Mehrfarbendarstellung mit 5 Zeilen Text; 16 Farben in verschiedenen Intensitätsstufen zuzüglich Schwarz = 121 Farbtöne

SCHNITTSTELLEN
Serielles Commodoreinterface, Steckleiste für eine ROM-Cartridge zur Speichererweiterung, achtpoliger Cassettenausgang, zwei Joystickbuchsen (achtpolig), Monitorausgang für Zusammensetzung/Farbe/Helligkeit/Audio, RF-Ausgang mit Hoch/Niedrig-Schalter, Netzanschluß (9 V)

PROGRAMMIERSPRACHEN
BASIC-3.5-Interpreter im ROM mit 75 Befehlen einschließlich Grafik

TASTATUR
66 Schreibmaschinentasten einschließlich 7 programmierbaren Funktionstasten und HELP

STÄRKEN
Hochentwickeltes BASIC, einfache Befehle für Tonsteuerung und Grafik, leichter Zugang zum Monitor für die Programmierung in der Maschinensprache.

werden. Ist die Frequenz eines Tones bekannt, müssen Sie nur im Handbuch den entsprechenden Wert für den Befehl nachschlagen.

SOUND 1,770,60

spielt den Ton A (mit der Frequenz von 440 Hz) für 60 Sechzigstel einer Sekunde auf Kanal 1.

Der tiefste Ton ist das A zwei Oktaven unter dem mittleren C (110 Hz) und der höchste das G zwei Oktaven über dem mittleren C (1575 Hz). – Das Gerät hat also einen Tonumfang von vier Oktaven. Es stehen dabei entweder zwei Musikkanäle (1 und 2) zur Verfügung oder ein Musikkanal und ein Kanal für weißes Rauschen (3). Da die Ausgabe in Mono erfolgt, werden beide Kanäle zu einem Signal zusammengefaßt, das nicht wieder getrennt werden kann.

Der Commodore 16 ist eine attraktive Maschine mit modernem BASIC und guten Grafikbefehlen. Seine Tonmöglichkeiten sind – selbst im Vergleich mit dem VC 20 – recht primitiv, wenn auch einfacher zu handhaben.

Das Softwareangebot ist aufgrund der „Nicht-Kompatibilität" zum C 64 und VC 20, für die eine riesige Programmbibliothek zur Verfügung steht, zur Zeit noch relativ gering.

Befehlsstrukturen

In dieser Folge fassen wir die Standardformate der Maschinencodeprogrammierung zusammen. Darunter fällt unter anderem die Adressierung des niedrigen und hohen Bytes.

Wenn Sie ein Programm mit RUN aufrufen, muß das Betriebssystem erst einmal feststellen, in welchem Bereich des Speichers der Programmtext überhaupt liegt. Es untersucht deshalb zuerst den Zeiger, der die Anfangsadresse des Programmtextes enthält. Dabei stellt sich die Frage, warum das Betriebssystem nicht statt der Pointeradresse die direkte Anfangsadresse speichert.

Der Hauptgrund für diese „Umständlichkeit" ist die Flexibilität. Sie erinnern sich: Das Betriebssystem ist ein im ROM festgelegtes Programm, dessen interne Daten (z. B. Speicheradressen) sich nicht ändern lassen. Nehmen wir an, daß über einen längeren Zeitraum mehrere Versionen eines Computers herausgebracht werden. Bei Version 1 genügte es vielleicht, den Anfang des Speicherbereiches für BASIC-Programmtexte auf die Adresse 2048 zu legen, für Version 2 aber wurde es notwendig, diesen Bereich bei Byte 4096 anfangen zu lassen. Ist eine Adresse jedoch erst einmal fixiert, dann können die neueren Maschinen nicht das Betriebssystem früherer Versionen einsetzen, da der Programmtext nicht an der alten Adresse beginnt. Für jede neue Maschine müßten daher teure, neue ROMs entwickelt werden, und außerdem würde Software, die auf einer Version entwickelt wurde, nicht auf anderen Modellen laufen. Enthält das Betriebssystem jedoch nur eine Zeigeradresse, dann läßt es sich für alle Versionen der Maschine verwenden, wobei nur der Inhalt des Pointers von Version zu Version geändert wird.

Es ist üblich, daß Zeigeradressen das sogenannte „lo-hi"-Format haben. Wenn z. B. Byte 43 und Byte 44 die Adresse 7671 enthalten (Seite 29, Offset 247), dann ist in Byte 43 der Wert 247 (Offset oder niederwertiges Byte der Adresse) gespeichert und in Byte 44 der Wert 29 (Seitenadresse oder höherwertiges Byte der Adresse).

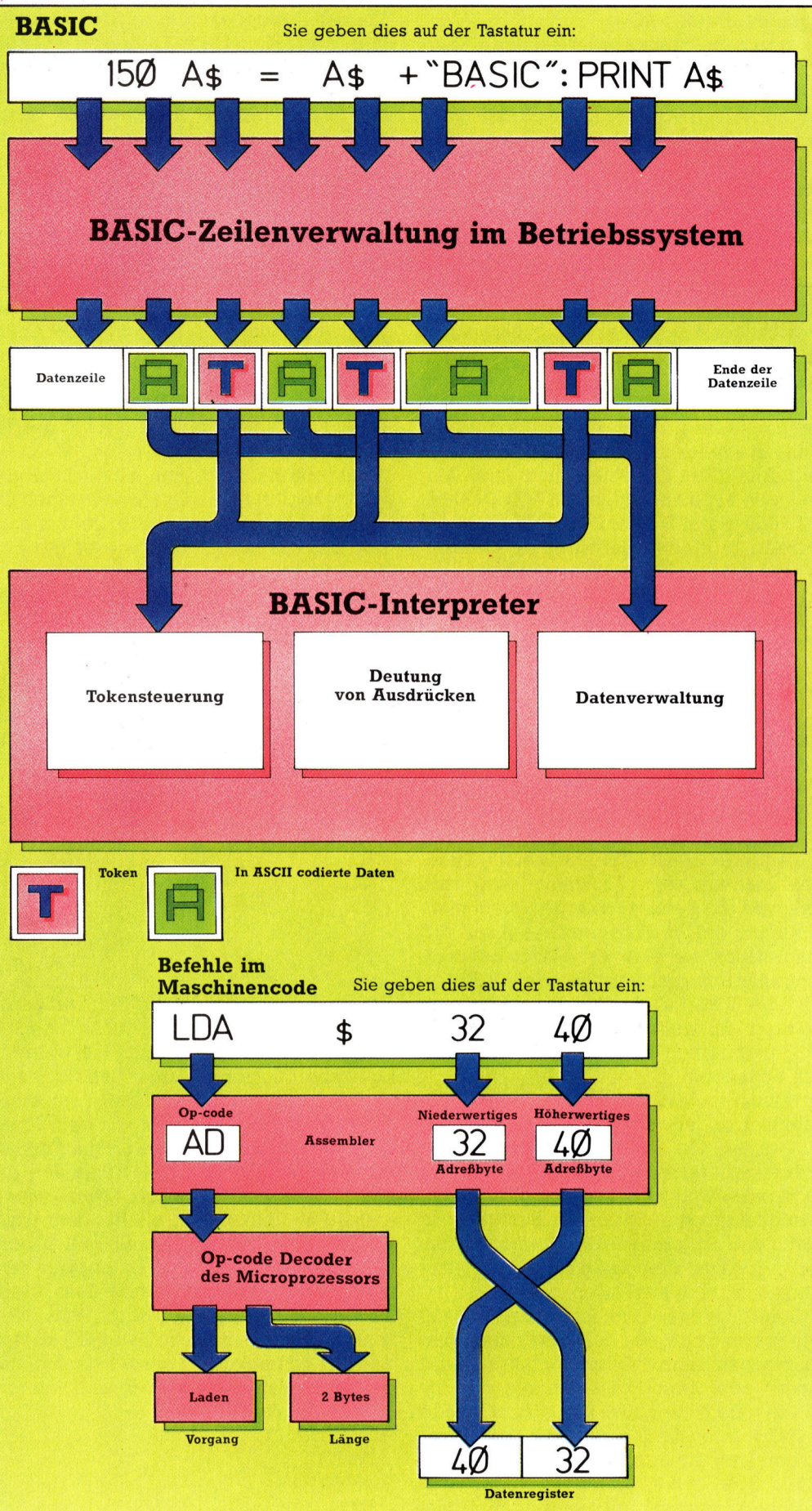
Bei einer Wiederholung des Beispiels mit Hexadezimalzahlen werden die Vorteile dieses Zahlensystems deutlich. (Von nun an werden wir Adressen und andere Zahlen im Hexadezimalformat schreiben. Hexzahlen sind dabei durch ein vorangehendes $ gekennzeichnet.) Der Zeiger besteht aus den Bytes $2B und $2C und zeigt auf die Adresse $1DF7. $2B enthält daher $F7 (das niederwertige Byte der Adresse), während $2C das höherwertige Adressbyte $1D speichert. In einer Hexadresse sind die beiden rechten Hexzahlen das niederwertige Byte, während die linken beiden Zahlen das höherwertige Byte (die Seitenadresse) darstellen.

Weiterhin bezieht man sich bei der Angabe einer Speicheradresse oft nur auf die Adresse, die das niederwertige Byte enthält. Man könnte daher sagen, daß auf dem Commodore 64 Byte 43 den Anfang des BASIC-Programmtextes angibt. Trotzdem ist es selbstverständlich, daß Byte 43 und Byte 44 gemeinsam der Pointer sind.

Token werden ebenfalls nach einem bestimmten Standard behandelt. Für die Programmierung im Maschinencode haben sie zwei Bedeutungen: Sie stellen englischsprachige Befehle wie PRINT oder RESTORE im numerischen Code mit einem Byte Länge dar und werden zudem als Offset eingesetzt. Obwohl die meisten BASIC-Befehle aus einem Wort bestehen, sind sie für das Betriebssystem nicht mit einem einzigen Vorgang erledigt. Der Befehl PRINT z. B. erfordert, daß Daten, die gedruckt werden sollen, im Speicher gefunden, bewertet und dann zeichenweise im ASCII-Code zum Bildschirm gesendet werden. Diese unterschiedlichen Aufgaben führt eine Unterroutine des BASIC-Interpreterprogramms aus. Findet der Interpreter in einer Programmzeile den Token für PRINT, dann sucht er mit diesem Wert die entsprechende Unterroutine und führt sie aus.

Einsatz des Tokens

Nehmen Sie an, in einer BASIC-Version gäbe es nur die drei Befehle INPUT, PRINT und STOP, denen jeweils die Token $80, $81 Unterroutinen des Interpreters fangen bei den Bytes $D010, $EA97 und $EC00 an. Diese Adressen sind im lo-hi-Format in sechs Bytes von $FA00 bis $FA05 gespeichert. Die Tabelle enthält also drei Zeiger mit einer Länge von je zwei Bytes. Wenn unser Interpreter jetzt einen Token findet, – z. B. $81 –, dann subtrahiert er davon $80, multipliziert den Rest mit 2 und addiert das Ergebnis zu $FA00. In diesem Fall ergibt das Endresultat $FA02, das das niederwertige Adressbyte des Zeigers der PRINT-Routine enthält. Der BASIC-Befehl PRINT wird also durch den Token $81 ersetzt. Dieser enthält den Offset für eine Zeigertabelle, mit der der Interpreter die eigene Unterroutine findet. Bei den anderen Token liefert dieser Algo-



Schritt für Schritt

Diese Darstellung verdeutlicht, wie eine Programmzeile in BASIC und ein Befehl im Maschinencode übersetzt und ausgeführt werden.

Das Betriebssystem gibt das Zeilenformat mit allen Informationen vor und ersetzt die BASIC-Schlüsselworte durch Token.

Sie geben RUN ein.

Der BASIC-Interpreter untersucht die Programmzeile nach Token und den entsprechenden Daten und ruft dann entsprechend dem Wert der Token die jeweiligen Unterroutinen auf.

Der Assembler übersetzt die mnemotischen Codes in den entsprechenden Op-code (1 Byte) und speichert den Operanden (2 Bytes) im lo-hi-Format.

Beim Ausführen des Befehls decodiert der Prozessor den Op-code in einen Operations- und einen Längencode, damit die richtige Anzahl der Bytes als Operand behandelt werden kann.

rithmus ebenfalls die Zeigeradresse der entsprechenden Unterroutine.

Hier wird der Unterschied zwischen BASIC, einer sogenannten „Hochsprache", und dem Maschinencode deutlich. BASIC ist für uns leichter zu verstehen, da es Codeworte enthält, die der englischen Sprache ähnlich sind, und außerdem algebraische Logik, Zahlen und Zeichenketten eingesetzt werden. Wenn wir die Schlüsselworte durch Token ersetzen und den Rest der Zeichen als ASCII-Codes darstellen, dann ähnelt ein Programm schon mehr dem Format, das ein Microprozessor verarbeiten kann.

Ein wichtiger Faktor bei Speicherabläufen ist das Konzept des „Zusammenhangs". Es gibt im Speicherbereich für BASIC-Texte eine ganze Reihe von Codes – ASCII-Codes zur Zeichendarstellung, Token als Kurzform für Befehle und spezielle Binärzahlen für Zahlendarstellung. Alle diese Codes erscheinen im Maschinencode als Binärzahlen von 00000000 bis 11111111 ($00 bis $FF oder 0 bis 255 dezimal). Sie werden in einem Speicherbyte untergebracht und nur aus ihrem Zusamenhang heraus interpretiert. Der Textbereich des Commodore 64 könnte z. B. folgende Programmzeile enthalten:

```
200 rem******left$********
```

Darin ist die Dezimalzahl 200 in drei verschiedenen Bytes enthalten: je einmal als niederwertiges Byte der Verbindungsadresse und der Zeilenadresse und einmal im Token für „left$". Jedes dieser Bytes sieht genauso aus wie die anderen, stellt aber etwas völlig anderes dar. Nur aus dem Zusammenhang wird deutlich, wie die Bytes zu interpretieren sind.

Am Anfang dieser Serie erwähnten wir, daß alle Information als eine Art Maschinencode im Computer untergebracht ist. Die ASCII-Codes und die Token sind uns nun vom Konzept her bekannt. Im weiteren Verlauf werden wir die Programmierung im Maschinencode ausführlich behandeln.

Programme im Maschinencode sind Bytefolgen, die sich an jedem Ort des Speichers befinden können. Sie enthalten eine Mischung aus Prozessorbefehlen und Daten, die der Prozessor verarbeiten soll. Wie bei anderen Speicherbytes lassen sich Daten von Befehlsbytes nur durch den Zusammenhang unterscheiden. Wir werden daher zunächst das Befehlsformat des Maschinencodes erklären.

Ein Befehl im Maschinencode beginnt immer mit einem Code, der den auszuführenden Vorgang genau angibt. Dieser Schlüssel wird „Op-code" oder „opc" genannt und hat die Länge von einem oder zwei Bytes. Ein Opcode kann völlig eigenständig sein. Im Normalfall folgen ihm aber ein oder zwei Datenbytes. Ein einzelnes Byte enthält dabei entweder eine numerische Konstante oder einen ASCII-Code, während zwei Datenbytes immer eine Adresse (Format: niederwertiges Byte/höherwertiges Byte) darstellen. Diese Definition läßt auch sofort die unterschiedliche Arbeitsweise der Microprozessoren deutlich werden: Der Acorn B wird von einem MOS-Tech-6502A gesteuert, der Commodore 64 von einem MOS-Tech-6510 (dem 6502A sehr ähnlich, wir sprechen daher nur von diesem), während in den Spectrum der Z80A von Zilog eingebaut ist. Der 6502 und der Z80 bauen auf dem gleichen Konzept auf, unterscheiden sich aber wesentlich in den Einzelheiten. Ganz besonders die Maschinencodes des Z80 und des 6502 sind völlig verschieden voneinander. So bestehen die Op-codes des 6502 immer aus einem Byte, das von einem, zwei oder keinem Datenbyte gefolgt werden kann, während Op-codes des Z80 zwei Bytes lang sein können, gefolgt von einem, zwei oder keinem Datenbyte.

Das interne Programm eines Microprozessors decodiert einen Op-code in einen Operations- und einen Längencode. Der Längencode gibt ihm die Information, wie er die auf den Op-code folgenden Bytes interpretieren soll. Nehmen wir die folgenden Hexadezimalbytes des 6502:

```
A9 0E 8D 01 4E 60 44 52 41 54
```

Darin sind drei Befehle enthalten, denen vier Bytes im ASCII-Code folgen. Die Bytes lassen sich auch so schreiben:

```
A9 0E
8D 01 4E
60
44
52
41
54
```

Der erste Befehl ist A9, auf den immer ein Datenbyte folgt. Der nächste Befehl ist 8D – immer gefolgt von zwei Datenbytes, während der Op-code 60 keine Daten benötigt, sondern eine Programmverzweigung auslöst, durch die die folgenden Datenbytes von dem Prozessor nicht mehr gelesen werden. Erhält der Microprozessor zuerst das Byte A9, laufen danach alle Befehle problemlos ab. Die in jedem Opc enthaltene Information stellt sicher, daß der Prozessor die richtige Anzahl von Datenbytes nach den Op-codes als Datenbytes versteht und daß das darauffolgende Byte wiederum als Op-code interpretiert wird. Wird dem Prozessor jedoch das zweite Byte (0E) übermittelt, wenn er einen Befehl erwartet, dann behandelt er dieses Byte als Op-code und interpretiert die Bytefolge so:

```
0E 8D 01
4E 60 44
52
```

Dem Opc 0E folgen zwei Datenbytes, ebenso dem Opc 4E, während Opc 52 keinen Op-code darstellt und einen Syntaxfehler verursacht. Damit wird deutlich, wie ein Mißverständnis am Programmanfang eine Kette von logischen Fehlern auslösen kann.

Unser Beispiel zeigt aber auch, wie unbequem Maschinencode (zumindest anfangs) zu lesen und zu schreiben ist. Die einzelnen Befehle unterscheiden sich nur durch ihre Stellung innerhalb des Speichers.

Mnemotische Zeichencodes

Einige dieser Unbequemlichkeiten lassen sich vermeiden, wenn für die Programmentwicklung mnemotische Zeichencodes statt numerischer Op-codes eingesetzt werden. Zum Laden der Programme müssen die Zeichen wiederum in Op-codes übersetzt werden. Die Assemblersprache eines Prozessors verwendet diese mnemotischen Codes, deren Übertragung in die Maschinensprache „Assemblierung" genannt wird. Zwischen einer Folge mnemotischer Zeichencodes und einem Set Op-codes besteht eine unmittelbare Übereinstimmung: Obwohl Assembler eine höhere Programmiersprache ist als der Maschinencode, ist der Unterschied zwischen beiden nur minimal.

Wenn wir die obige Maschinencodefolge in der Assemblersprache des 6502 darstellen, sieht das folgendermaßen aus:

```
0000 A9 0E     LDA #$0E
0002 8D 01 4E  STA $4E01
0005 60        RTS
```

Im Assembler des Z80 sieht die gleiche Folge jedoch völlig anders aus:

```
0000 3E 0E     LD A, $06
0002 32 01 4E  LD($4E01),A
0005 C9        RET
```

Die erste Spalte zeigt dabei die Hexadezimaladresse des ersten Bytes in der Zeile an – der Opc A9 des 6502 liegt dabei in Byte 0, während das Seitenbyte 4E sich bei beiden Listen in Byte 4 befindet etc. Die zweite Spalte kann ein, zwei oder drei Bytes enthalten. Sie zeigt den Maschinencode an. Die dritte Spalte beginnt mit den mnemotischen Codes der Assemblersprache und stellt die Assembler-Version des Maschinencodes dar. Versuchen Sie nicht, die Befehlsfolge zu verstehen. Für den Augenblick genügt es, daß Sie die Assemblerliste und die Unterschiede zwischen den Versionen des Z80 und des 6502 erkannt haben. Achten Sie dabei auch darauf, daß die Adresse der zweiten Zeile im Maschinencode im lo-hi-Format erscheint, während sie im Assembler das „normale" hi-lo-Format annimmt.

In unserer nächsten Folge werden wir mit der ausführlichen Untersuchung der einzelnen Opcodes anfangen.

Hexadezimalanzeige

In den Mempeek-Programmen läßt sich eine Hexadezimaldarstellung statt der dezimalen Anzeige mit folgenden Änderungen erreichen:

Acorn B

Fügen Sie folgendes hinzu:

```
3000 DEF PROCHXPRINT(DECNUM)
3100 LOCAL X$
3200 X$="0123456789ABCDEF"
3300 HB=INT(DECNUM/16):LB=DECNUM—HB*16
3400 B$=MID$(X$,HB+1,1)+MID$(X$, LB+1,1)+" "
3500 PRINT B$;
3600 ENDPROC
```

und ändern Sie Zeile 600 in:

```
600 PROCHXPRINT(PK%)
```

Spectrum

Fügen Sie folgendes hinzu:

```
  10 LET X$="0123456789ABCDEF"
3000 REM******HEX BYTE S/R*******
3100 LET HB=INT(PK/16):LET LB=PK-HB*16
3200 LET B$=X$(HB+1)+X$(LB+1)+" "
3300 PRINT B$;
3400 RETURN
```

und ändern Sie Zeile 600 in:

```
600 GOSUB 3000
```

Commodore 64

Fügen Sie folgendes hinzu:

```
  10 LET X$="0123456789ABCDEF"
3000 REM******HEX BYTE S/R*******
3100 HB=INT(PK/16):LB=PK-HB*16
3200 B$=MID$(X$,HB+1,1)+MID$(X$,LB+1,1)+" "
3300 PRINT B$;
3400 RETURN
```

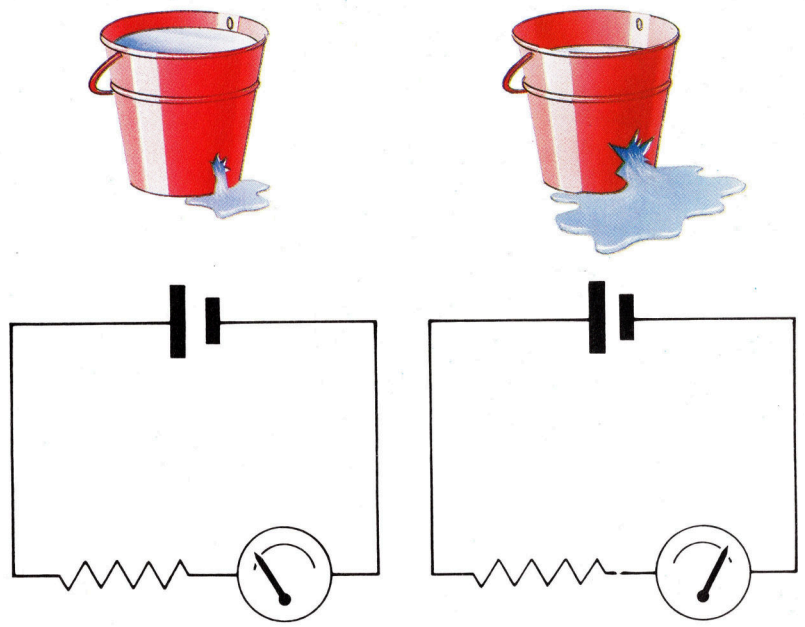
und ändern Sie Zeile 600 in:

```
600 GOSUB 3000
```

Die Änderungen veranlassen, daß die Speicherinhalte im Hexadezimalformat angezeigt werden. Die Anfangsadresse und die Anzahl der Bytes müssen jedoch in Dezimalzahlen angegeben werden.

Strom und Spannung

Elektrizität läßt sich am besten über ihre Basisgrößen Strom, Spannung und Widerstand mathematisch erklären.

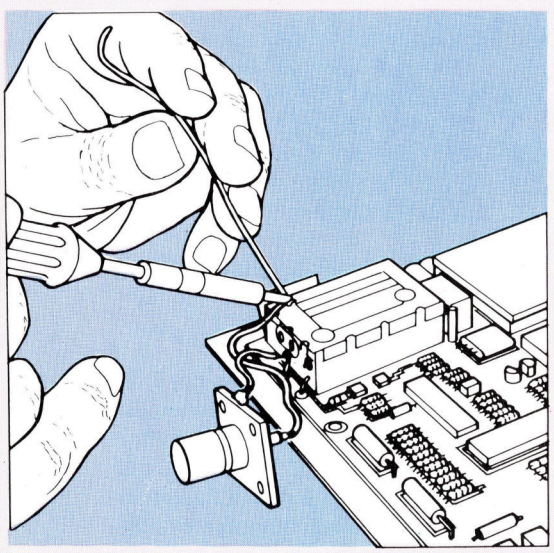


Strom und Widerstand

Die Bauteile in einem elektrischen Stromkreis setzen dem Stromfluß einen gewissen Widerstand entgegen, vergleichbar mit der Wirkung der Ventile und Rohre eines Heizungskreislaufes: Je größer der Widerstand ist, um so weniger Wasser fließt. Durch das Loch in einem Eimer läuft um so weniger Wasser, je kleiner das Loch ist. Wenn in einem Stromkreis ein hoher elektrischer Widerstand liegt, fließt nur ein geringer Strom. Verringert man den Widerstand, wird der Stromfluß größer. Der elektrische Widerstand eines Bauteils hängt hauptsächlich vom verwendeten Material ab – Leitungsdrähte etwa werden aus Kupfer hergestellt, weil es dem Strom nur geringen Widerstand entgegensetzt.

Klare Sicht

Nach einem einfachen Umbau kann der Sinclair Spectrum ein Videosignal liefern, mit dem sich ein hochauflösender Farbmonitor ansteuern läßt. Der Bild-Modulator ist als größerer silberner Kasten (oben links auf der Platine) leicht erkennbar. Am hinteren der beiden Anschlüsse des Modulators wird eine kurze Leitung angelötet, die Sie mit dem Außenkontakt einer BNC-Buchse verbinden. Den Mittelkontakt der Buchse schließen Sie an das Gehäuse des Modulators an.



In der letzten Folge unseres elektronischen Bastelkurses wurden einige Grundlagen der Elektrotechnik dargelegt. Doch in welcher Beziehung stehen die einzelnen Größen zueinander und wie kann man sie berechnen? Der deutsche Physiker Georg Ohm (1789–1854) hat als erster den Zusammenhang zwischen Strom, Spannung und Widerstand untersucht. Ohm erkannte, daß sich aus zweien dieser Größen jeweils die dritte errechnen läßt: Die Spannung ergibt sich, indem der Widerstand mit dem Strom multipliziert wird. Diese Gleichung ist aufgrund der einfachen Arithmetik sehr leicht umzustellen, und schon läßt sich mit einer Division des Stromwertes durch die Spannung der Widerstand bzw. durch Teilen der Spannung durch einen Widerstandswert die dazugehörige Stromstärke ermitteln.

Das Ohm'sche Gesetz

In der Elektrotechnik werden diese Größen oft symbolisch geschrieben. Dabei steht U für die Spannung, I für den Strom und der Buchstabe R für den Widerstand. Ohms Gleichungen sehen dann so aus:

$$U=RxI$$
$$R=U/I$$
$$I=U/R$$

Zum Ausrechnen der Leistung P wird noch die Formel

$$P=Ux\ I$$

gebraucht. Sie gibt den Stromverbrauch einer Schaltung bzw. eines Gerätes in Watt an, der sich aus der anliegenden Spannung multipliziert mit dem fließenden Strom ermitteln läßt.

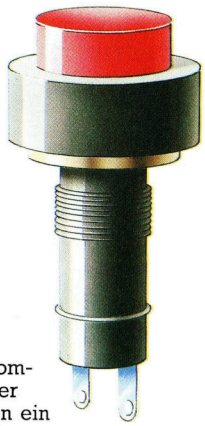
Die Gleichung für die Leistung können Sie auch im Haushalt einsetzen, wenn Sie zum Beispiel den Wert einer neuen Sicherung berechnen wollen: Bei einer Waschmaschine mit einem Verbrauch von 3 kW bei 220 Volt rechnen Sie so: 3000 (Leistung in Watt) geteilt durch 220 (Netzspannung in Volt) ergibt einen Wert von 13,64 Ampere für den Strom. Sie müssen also eine Sicherung mit 16 Ampere einsetzen – das ist der nächsthöhere erhältliche Wert. Der betreffende Stromkreis kann dann – außer mit der Waschmaschine – noch mit insgesamt 520 Watt belastet werden (16 A – 13,64 A = 2,36 A x 220 Volt). In der nächsten Folge zeigen wir, wie Sie die Ohm'schen Gesetze experimentell nachvollziehen können.

ACHTUNG: Der Hersteller kann die Garantieleistung nach dem unbefugten Öffnen des Gehäuses verweigern!

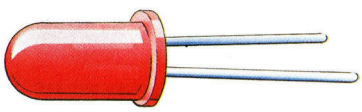
Bauteile

Für den Laien erscheint ein elektronischer Schaltplan recht kompliziert. Dabei kann die Darstellung eines Gerätes durch genormte Zeichen oft viel Arbeit sparen, weil sie Funktion und verwendete Bauteile übersichtlicher macht. Als Beispiel ist unten der Schaltplan einer Sprechanlage abgebildet. Daneben finden Sie die einzelnen Bauteile mit einer kurzen Beschreibung ihrer Funktionen. Auf Schaltplänen sind die Bauelemente meist mit einer codierten Bezeichnung – etwa „R1" oder „TR 2" – versehen. Die Verbindungslinien im Plan stellen die Leitungen dar – im Gerät sind es entweder Drähte oder Leiterbahnen auf einer Platine.

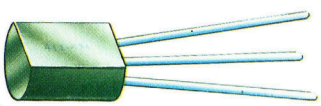
1 Schalter
Schalter gibt es in vielen verschiedenen Ausführungen, die zudem noch unterschiedliche Funktionen haben. Der Kontakt kann dauerhaft erhalten bleiben oder sich beim Loslassen wieder lösen.



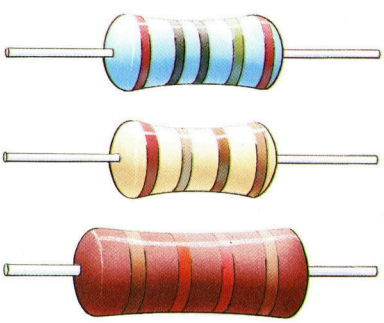
2 Leuchtdioden
Dioden sind die unkompliziertesten Halbleiter und stellen sozusagen ein „elektronisches Ventil" dar: Ein Strom kann sie nur in einer Richtung durchfließen. Manche Dioden haben kein Metall-, sondern ein durchsichtiges Kunstharz-Gehäuse. Bei Stromdurchgang geben diese „LEDs" Licht ab.

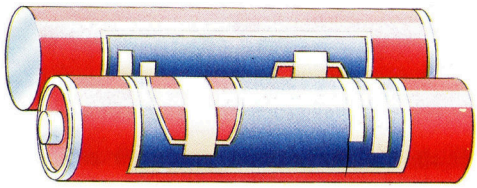


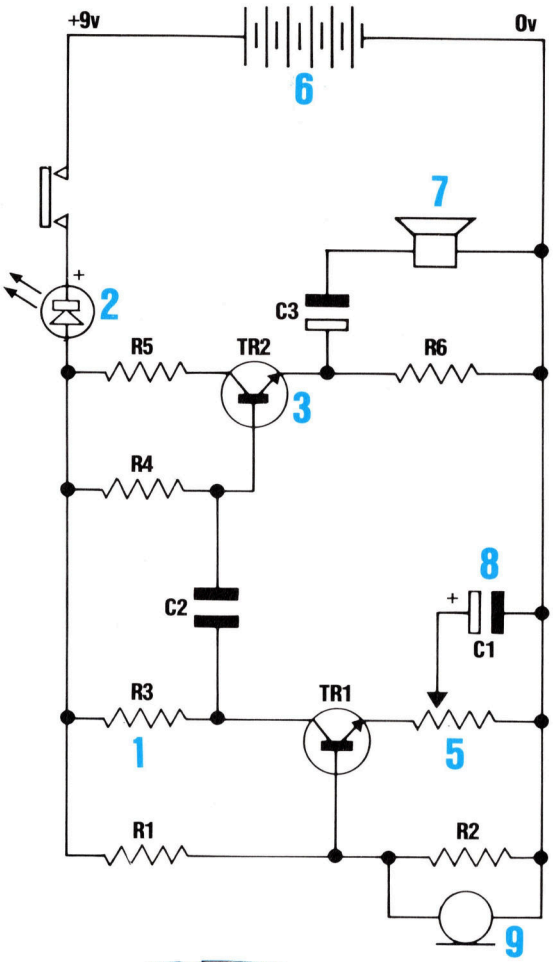
3 Transistoren
Je nach Konstruktion und Anschlußweise kann ein Transistor Schalter oder auch Verstärker sein. Die Erfindung des Transistors im Jahre 1947 leitete das Zeitalter der Microelektronik ein.

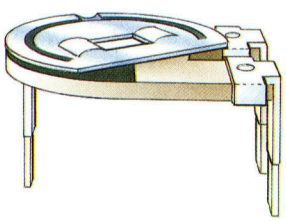


4 Widerstände
Durch Einbau von Material geringerer Leitfähigkeit wird der Stromfluß in einer Schaltung reguliert. Dazu dienen Widerstände in verschiedenen Größen, deren Widerstandswert sich an den farbigen Ringen ablesen läßt.





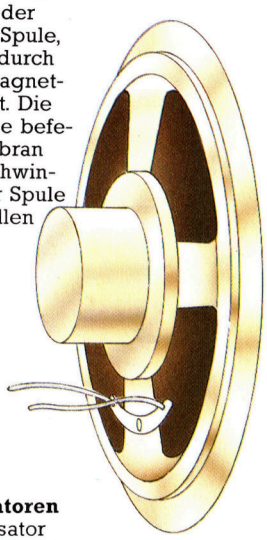




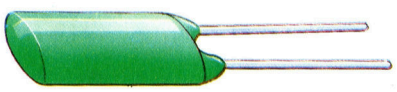
5 Einstellbare Widerstände
Widerstände, bei denen sich die Leitfähigkeit regulieren läßt, werden „Potentiometer" oder „Trimmer" genannt. Meist wird zum Einstellen ein Schleifer über eine Kohlebahn bewegt. Je nach Stellung des Schleifers wird der Widerstand größer oder kleiner.

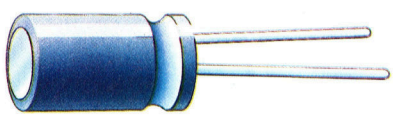
6 Batterien
Kleinere Schaltungen lassen sich am günstigsten mit Batterien versorgen, weil sie eine sehr stabile Gleichspannung liefern.

7 Lautsprecher
Lautsprecher sind in ihrer Funktionsweise Mikrofonen sehr ähnlich. Im Lautsprecher durchfließt der Strom eine Spule, die sich dadurch in einem Magnetfeld bewegt. Die an der Spule befestigte Membran setzt die Schwingungen der Spule in Schallwellen um.

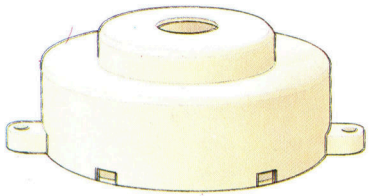


8 Kondensatoren
Ein Kondensator kann elektrische Ladungen speichern. Wenn die Anschlüsse mit einer Stromquelle verbunden werden, lädt sich der Kondensator auf. Er wird erst wieder entladen, wenn sich die Anschlüsse berühren oder durch einen externen Schalter miteinander verbunden werden.





9 Mikrofone
In einem Mikrofon werden die Schallwellen der Luft von einer feinen Membran aufgefangen. Eine damit verbundene Spule erzeugt bei ihren Bewegungen im Magnetfeld eine Spannung.



Von Göttern und Menschen

Das Abenteuerspiel „Valhalla“ zeigt schon im Namen seinen Bezug zur nordischen Mythologie. Die hervorragende Grafik fasziniert.

Valhalla läuft auf dem Spectrum und auf dem Commodore 64. Immerhin 81 verschiedene „Schauplätze“ gibt es in diesem Abenteuerspiel – 16 davon in Asgard (einschließlich der himmlischen Burg Valhalla), 20 in der Menschenwelt Midgard. Die übrigen 45 Plätze bilden gemeinsam die Unterwelt. Jeder Ort hat acht Ausgänge, die jedoch versperrt sind bzw. nur im Besitz magischer Gegenstände durchschritten werden können. Die meisten Szenarios sind durch „Ringwege“ miteinander verbunden – wenn er den geeigneten Ring trägt, kann der Spieler größere Entfernungen überspringen und somit sein Ziel schneller erreichen.

Die 36 im Spiel auftretenden Personen sind entweder freundlich oder böse. Mindestens drei von ihnen befinden sich jeweils auf dem Bildschirm, wo sie Kämpfe austragen, sich bewirten oder auch mit Gegenständen nach ungebetenen Gästen (oder passiven Spielern) werfen.

Sie können sich in die Auseinandersetzungen einmischen oder einfach nur zusehen. Allerdings kommt der „stille Beobachter“ über kurz oder lang zu Tode!

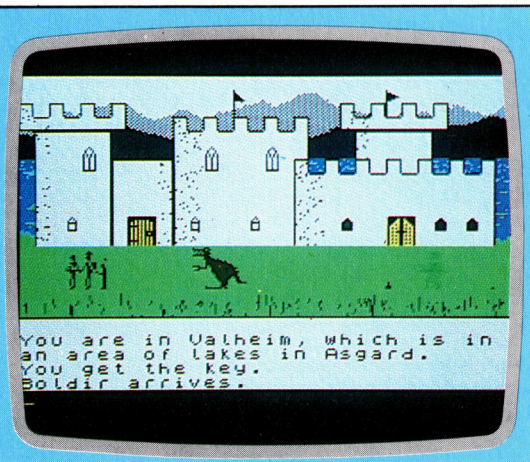
Den Götterwohnsitz Valhalla kann der Spieler auf unterschiedlichen Wegen erreichen. Dazu muß er allerdings die Unterstützung freundlicher Helfer gewinnen. Wer sich aber lieber auf die Seite der Bösewichte schlägt, kann bei ausdauernder Schlechtigkeit auf die Hilfe von Mächten der Unterwelt bauen.

Speziell bei der Commodore-Version sind die grafischen Darstellungen im Spiel sehr gut. Anfangs erscheint der Hintergrund, der beim Spectrum in kräftigen Farben, beim Commodore 64 in sanften Pastelltönen gehalten ist. Beim Spectrum sind die Personen schwarz und ähneln Strichmännchen, die feinere Grafik- und Farbauflösung des Commodore zeigt sie um einiges attraktiver. Zum Schluß erscheinen die verschiedenen Gegenstände, die sich an dem jeweiligen Ort befinden, auf dem Bildschirm: Das können etwa Schlüssel, Waffen oder wertvolle Juwelen, aber auch Wegzehrung in Form von Brot oder Wein sein. Welche Handlungen die auftretenden Figuren ausführen, wird unten auf dem Bildschirm mit Worten beschrieben. Hier greift der Spieler ein, wobei die Liste möglicher Aktionen recht umfangreich ist: Sollte etwa Ihre Überzeugungskraft nicht ausreichen, einen der nordischen Götter zum Teilen seines Schatzes zu überreden, darf der Versuch auch mit Waffengewalt fortgesetzt werden. Auf der Suche nach dem sagenhaften Valhalla ist häufig der Besitz eines magischen Gegenstandes hilfreich – nicht immer sind diese „Schlüssel zum Paradies“ ganz einfach zu bekommen.

Valhalla: Verfügbar für den 48K-Spectrum und den Commodore 64
Hersteller: Legend, Freepost, 1 Milton Road, Cambridge CB4 1 UY, (GB)
Autoren: Graham Asher, Richard Edwards, Charles Goodwin, James Learmont, Jan Ostler, Andrew Owen, John Peel
Joysticks: Nicht notwendig
Format: Cassette

Tag und Nacht

Valhalla ist eines der Spiele, bei denen das Verstreichen der Zeit durch allmähliches Abdunkeln des Bildschirmes angedeutet wird: So entsteht der Eindruck von Tag und Nacht.



Neue Ausdrucksform

Die heute mögliche digitale Codierung von Klängen – Sampling genannt – erlaubt dem Musiker die Erzeugung jedweder Art von akustischen „Farben".

Alannah Currey, Tom Bailey und Joe Leeway von „Thompson Twins" (von vorn nach hinten). Die Gruppe bestand ursprünglich aus einer siebenköpfigen Formation, arbeitet heute aber als Trio mit vorprogrammierten, auf Band aufgezeichneten Rhythmen, die die Original-Rhythmusgruppe ersetzen.

Trotz der zahlreichen Entwicklungen in diesem Bereich sollten wir uns die Funktionsweise eines einfachen, spannungsgesteuerten Oszillators vergegenwärtigen. Wenn ein körperlicher Gegenstand – sei es der Flügel einer Biene oder das Stimmband des Menschen – vibriert, wird die umgebende Luft sehr schnell zusammengedrückt, und sie dehnt sich aus. Dabei wird eine Welle erzeugt, die vom menschlichen Ohr eingefangen und dann vom Gehirn als Geräusch interpretiert wird. Schickt man eine elektrische Spannung durch einen Modulator auf einen Metallstreifen, beginnt dieser zu vibrieren. Dabei entsteht die einfachste Form einer Welle, die sogenannte Sinuswelle. Die Tonhöhe oder Frequenz der Vibration hängt von der zugeführten Stromstärke und von der Dichte des verwendeten Leiters ab, in unserem Fall also des Metallstreifens. Diese kleine tonerzeugende Einheit nennt man Oszillator. Über Jahrzehnte war dies eine der wenigen Grundmethoden zur Erzeugung synthetischer Musik.

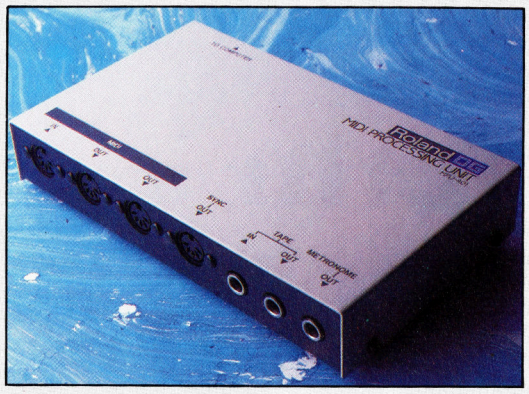
Das erstmals 1983 auf den Markt gekommene MIDI-Interface ist eine Einheit, die speziell dafür entwickelt wurde, mit einem Digitalsystem (beispielsweise einem Computer) ein anderes zu kontrollieren bzw. zu steuern, z. B. einen Synthesizer.

Unterschiedliche Systeme

Jahrelang waren Studios mit unterschiedlichsten Klangverarbeitungssystemen ausgestattet. Die Anzahl der Filter- und Hallanlagen galt als Maßstab für die Qualität eines Studios. In den siebziger Jahren waren auch Synthesizer-Spieler bei Bühnenauftritten von etlichen Keyboards umgeben, die ganz individuelle Einstellungen erforderten.

Machen wir uns klar, was in einem gut ausgestatteten Aufnahme-Studio geschieht, indem wir an ein bekanntes Gesellschaftsspiel denken. In diesem Spiel flüstert ein Spieler dem nächsten einen Satz zu und so weiter. Der letzte Spieler sagt dann natürlich auch den weitergeleiteten Satz, diesmal allerdings laut. Und hier ist interessant zu hören, was vom ursprünglichen Satz übriggeblieben ist. Eine ganz klare Aussage kann in eine Ansammlung sinnloser Wörter umgewandelt worden sein oder umgekehrt. Ähnliches geschieht in einem Aufnahmestudio, nur daß es sich hier beim Original-„Satz" um eine Folge musikalischer Klänge handelt. Und statt der Reihe von Zuhörern, die jeweils eine neue Version des Gehörten produzieren, steht dort eine Reihe von klangbearbeitenden Geräten zur Verfügung, die alle einzeln kontrollierbar sind und eine spezielle Aufgabe zu erfüllen haben. Ein erfahrener Tontechniker kann jede Einheit in wenigen Sekunden genau einstellen, doch welche Probleme bei der Synchronisation entstehen, ist leicht vorstellbar.

Die wichtigste Entwicklung der siebziger Jahre war die interne Klangerzeugung eines Synthesizers. Bestes Beispiel für diese Entwicklung ist der tastaturgesteuerte Baß-Synthesizer. In den sechziger Jahren basierte die Popmusik, damals der Soul, auf dem elektrischen Baß. Der Sound wurde als Motown-Stil bekannt. Ab etwa 1970 erreichten Funk-Bassi-



Kontrollzentrum
Das MP-401-MIDI-Interface von Roland ist ein perfekter Baustein zur Verbindung von Microcomputern mit digitalen Synthesizern. Es kontrolliert alle externen Synchronisierungsfunktionen, das interne wie externe Timing, Band-In- und -Output und den Synthesizer-Output. Das bedeutet: Der Computer hat lediglich die Funktion, Befehle zu senden und zu empfangen und verwaltet den Speicher. Das MP-401 kostet etwa 600 Mark.

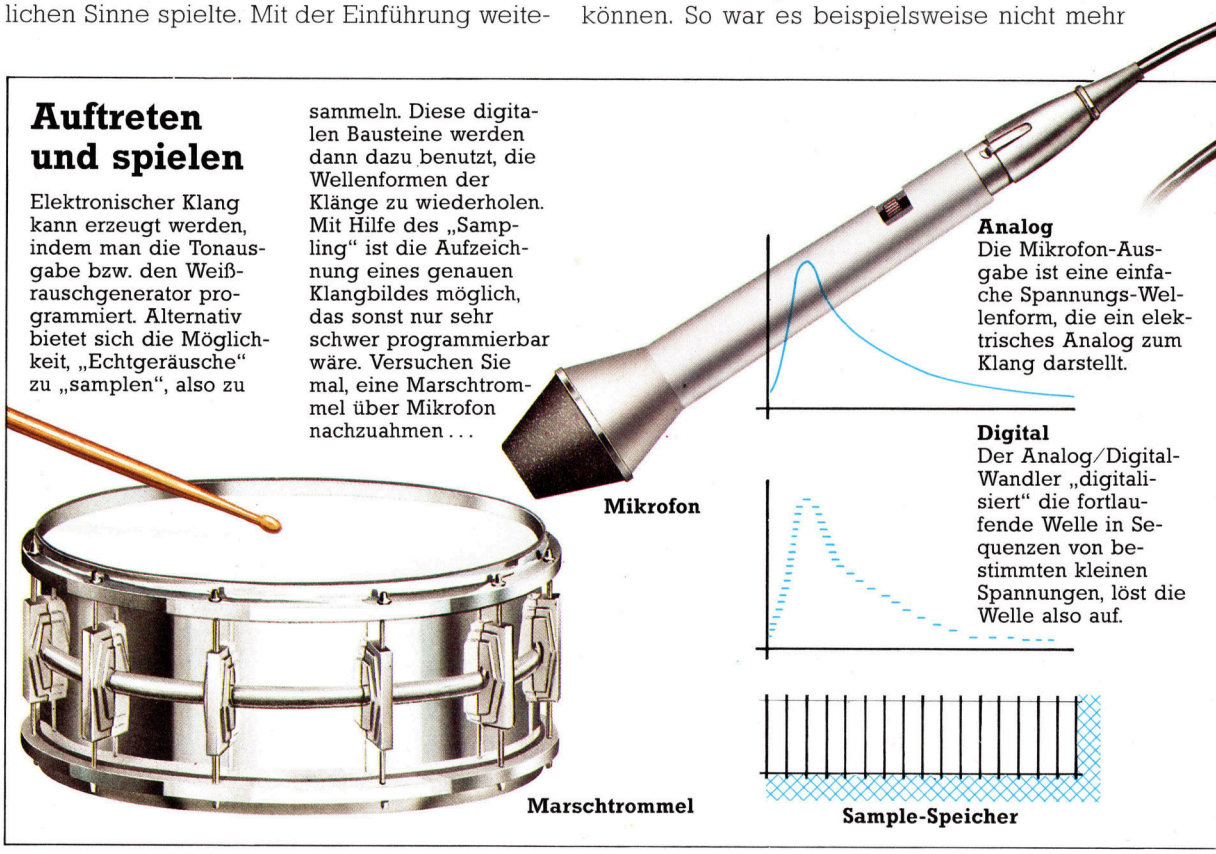
sten eine Virtuosität auf ihren Instrumenten, die denen der Lead-Gitarristen gleichkam. Am Ende des Jahrzehnts kam jedoch der tastaturgesteuerte Baß-Synthesizer auf den Markt. Daraus ergaben sich für den Keyboardspieler neue Probleme, denn er hatte nun den Part des Baß-Spielers zu übernehmen. Das bedeutete: Er mußte mit dem Drummer zusammenspielen, um ein genau getimtes rhythmisches Grundgerüst zu schaffen. Das wiederum bedeutete, daß er keine „Keyboard"-Musik im eigentlichen Sinne spielte. Mit der Einführung weiterer Analog-Synthesizer konnte man auf dem Keyboard Trompete, Saxophon und Schlagzeug spielen. Immer mehr Keyboarder benutzten ein im Prinzip einfaches Zusatzgerät, um diese „Instrumente" in den Griff zu bekommen: den Sequenzer.

Ein Sequenzer ist ein Gerät, das mehrere Bestandteile einer Komposition nach einem bestimmten Muster miteinander vereint. Sein Funktionsprinzip: Kontrollierte Spannungen werden in einen Oszillator eingegeben. Je höher die Spannung, desto schneller vibriert der Metallstreifen. Die sich daraus ergebende Wellenform ist als sehr hoher Ton hörbar. Der Oszillator wird mit einem Sequenzer kontrolliert. Das ist erforderlich, da Musik nur in den seltensten Fällen aus fortlaufenden Klangabfolgen besteht. Zur Erzeugung rhythmischer Muster und genereller musikalischer Strukturen sind kurze Breaks oder lange Pausen erforderlich. Ein „Break" entsteht in einem Klangmuster dadurch, daß eine Nullspannung in den Oszillator gespeist wird. Sinn dieses „Sequenzens" ist sicherzustellen, daß der Break jedesmal an derselben Stelle erfolgt, wenn das Muster wiederholt wird.

Kaum ein Synthesizer der siebziger Jahre verfügte über Sequenzer-Möglichkeiten. Doch die Musiker wußten bald, was mit dem Verfügbaren machbar war. Die hämmernde Disco-Musik, die Giorgio Moroder mit Donna Summer produzierte, ist Sequenzer-Musik schlechthin. Die neueren Synthesizer-Gruppen entwickelten einen völlig neuen Stil, um die technischen Möglichkeiten voll ausnutzen zu können. So war es beispielsweise nicht mehr

Auftreten und spielen

Elektronischer Klang kann erzeugt werden, indem man die Tonausgabe bzw. den Weißrauschgenerator programmiert. Alternativ bietet sich die Möglichkeit, „Echtgeräusche" zu „samplen", also zu sammeln. Diese digitalen Bausteine werden dann dazu benutzt, die Wellenformen der Klänge zu wiederholen. Mit Hilfe des „Sampling" ist die Aufzeichnung eines genauen Klangbildes möglich, das sonst nur sehr schwer programmierbar wäre. Versuchen Sie mal, eine Marschtrommel über Mikrofon nachzuahmen ...



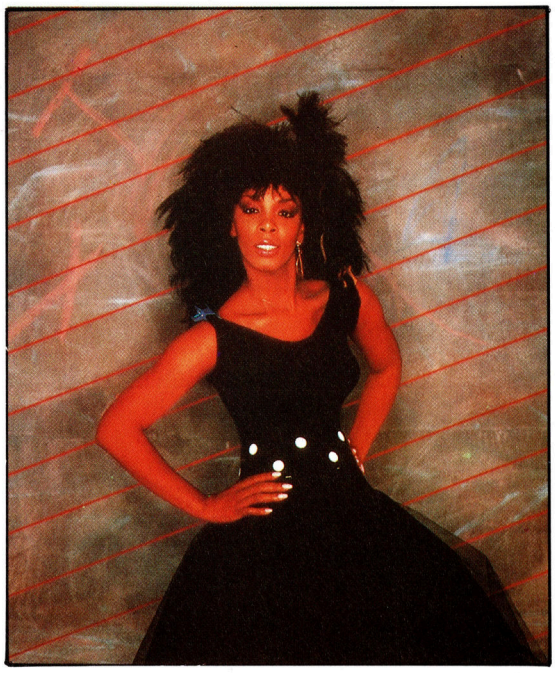
nötig, jede einzelne Note zu spielen, wie es noch Rick Wakeman getan hatte. Statt dessen konnte eine ganze Sequenz durch Veränderung einer einzigen Einstellung gestartet oder gestoppt werden. In der Zwischenzeit konnte der Spieler sein Keyboard verlassen und im exakten Timing zum Rhythmus des Sequenzers tanzen, dann an ein anderes Keyboard gehen und z. B. eine Melodie oder eine Akkordfolge spielen. Mitte der achtziger Jahre wurde der Bühnenstil von Gruppen wie den „Thompson Twins" nahezu völlig durch den Synthesizer geprägt.

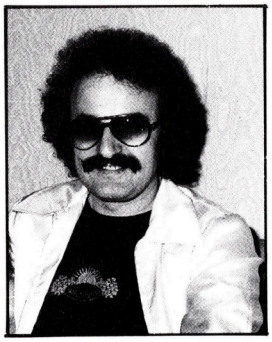
Klang-Digitalisierung

Die ersten digitalen Synthesizer waren nach dem Vorbild ihrer analogen Vorläufer konstruiert worden. Bestes Beispiel dafür ist die Beliebtheit der Linn-Rhythmus-Maschine, einer der ersten Klangerzeuger, bei denen Sample-Sounds verwendet wurden. Neben der reinen Aufzeichnung von Musik gibt es, bedingt durch den Videoboom der letzten Jahre, weitere neue Aufgaben für die Studios. Video hat einen neuen Stil und ein neues Bewußtsein bei der Imagepflege geschaffen. Daraus resultiert die Forderung der Produzenten, daß die Musik das auch „rüberbringt".

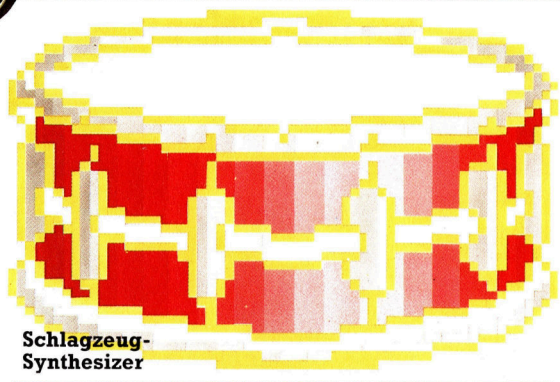
Wird die Video-Begleitmusik mit herkömmlichen Instrumenten erzeugt, ist der Synchronisierungsvorgang mit der Technik vergleichbar, die auch beim Film Anwendung findet. Wird jedoch die Musik durch auf Digital-Synthesizer-Basis erzeugte individuelle Klänge und Sequenzen eingebracht, gibt es viele Fehlerquellen. Nehmen wir einmal an, daß in einem Videofilm eine Blumenvase von einem Tisch fällt und zerbricht. Der Musiker muß eine Klangsequenz schaffen, die dort endet, wo die Vase herunterfällt und dann einen Percussion-Klang für den Augenblick einfügen, in dem die Vase auf dem Boden zerschellt. Die Aufnahme beginnt und die Sequenz läuft. Doch bald zeigt sich, daß die Sequenzdauer falsch berechnet wurde. Die Sequenz endet, obwohl die Vase noch auf dem Tisch steht. Der Musiker versucht nun, den Percussion-Klang einzubringen, der auf einer anderen Spur aufgezeichnet ist. Die Aufnahme wird neu gestartet, doch diesmal wird das Geräusch des Zerbrechens einen Sekundenbruchteil zu spät aufgezeichnet. Musiker brauchen also eine Möglichkeit, digitale Instrumente so miteinander zu verbinden, daß alles zur rechten Zeit geschieht. Erforderlich ist also ein digitales Interface.

Ähnliche Probleme hat ein Synthesizer-Spieler beim Live-Auftritt. Seine Ausrüstung beinhaltet zwei digitale Synthesizer von unterschiedlichen Herstellern und eine Linn-Rhythmus-Maschine. Auf dem einen Synthesizer und auf der Linn laufen Sequenzen, doch sie sind nicht absolut synchronisiert. Das endet meist so: Der Musiker läßt die Linn automatisch weiterlaufen und spielt den anderen Synthi manuell weiter. Ergebnis: Der zweite Synthesizer, eigens wegen seiner vorprogrammierbaren Klänge erworben, bleibt unbenutzt. Der Musiker benötigt also ein Bindeglied zwischen den einzelnen Instrumenten, um das vorproduzierte Klangmaterial an der richtigen Stelle zur richtigen Zeit einsetzen zu können. Außerdem sollte der Sequenzer des ersten Synthesizers die vorprogrammierten Klänge des zweiten spielen können. Für diesen Zweck ist MIDI genau das Richtige.

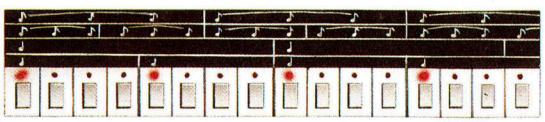




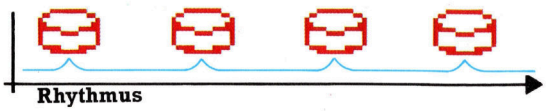
Donna Summer und ihr Produzent Giorgio Moroder gehörten zu den ersten Popmusikern, die vorwiegend elektronisch erzeugte Rhythmen und Synthesizer bei Plattenaufnahmen verwendeten.



Schlagzeug-Synthesizer



Elektronischer Trommelschlag
Mit Hilfe des Synthesizers kann jeder Trommelrhythmus konstruiert werden (jede Taste entspricht einem bestimmten Schlag). Beim Playback wird das Keyboard durch ein vorprogrammiertes Signal eingeschaltet.



Finale Grande

Das Projekt LOGO-Adventure nähert sich dem Ende. Nachdem die einzelnen Räume sowie die Bewegungsabläufe definiert wurden, geht es jetzt darum, dem Spiel durch Überraschungseffekte und gefährliche Situationen Spannung zu verleihen.

Zu diesem Zeitpunkt enthält „Zoltoths Schrein" nur zwei Gefahren. In Raum.4 erwartet den Spieler eine unfreundliche Schlange. Das Programm ruft nun die Gefahren-Prozedur auf:

```
TO SCHLANGE.GREIFTAN
  PRINTL [[EINE MAECHTIGE SCHLANGE]
    [KOMMT DIREKT AUF DICH ZU!]]
END
```

Die zweite Gefahr wirkt sich zwar nicht unmittelbar auf das Geschehen aus, kann jedoch später unliebsame Folgen nach sich ziehen:

```
TO TOR
  PRINTL [[EIN GROSSES GOLDENES TOR
    SCHLIESST SICH HINTER DIR] [DER
    SUEDLICHE AUSGANG IST
    VERSPERRT]]
  MAKE "GEFAHREN[]
  MAKE "AUSGANG.LIST [[N 7] [O 8]]
END
```

Nun soll die NEHMEN-Prozedur so geändert werden, daß der Spieler den Ring nicht aufnehmen kann, solange er das Schwert trägt.

```
TO AUFNEHMEN :ITEM
  IF :ITEM = "RING THEN NEHMEN.RING
    STOP
  RECHNE.ZU.INV :ITEM
  ENTFERNE.AUS.RAUM :ITEM
END

TO NEHMEN.RING
  IF MEMBER? "SCHWERT :INVENTAR
    THEN PRINT [DU KANNST DEN RING
    NICHT AUFHEBEN] STOP
  RECHNE.ZU.INV :ITEM
  ENTFERNE.AUS.RAUM :ITEM
END
```

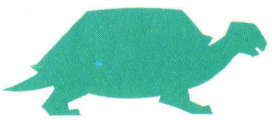
Prüfroutinen

Die folgenden Routinen bewirken, daß der Spieler den aufgenommenen Gegenstand untersuchen kann.

```
TO UNTERSUCHEN :OBJ
  IF :OBJ = "RING THEN RING.PRUEF
    STOP
  IF :OBJ = "TRUHE THEN TRUHE.PRUEF
    STOP
  IF :OBJ = "SCHWERT THEN
    SCHWERT.PRUEF STOP
  PRINT [DU SIEHST NICHTS
    BESONDERES]
END

TO RING.PRUEF
  IF HIER? "RING THEN PRINTL [[AUF DEM
    RING BEFINDEN SICH UNLESBARE
    ZEICHEN:] [R—— —E]] ELSE PRINT
    [ICH SEHE KEINEN RING]
END

TO HIER? :OBJ
  IF MEMBER? :OBJ :OBJEKTE THEN
    OUTPUT "TRUE IF MEMBER? :OBJ
    :INVENTAR THEN OUTPUT "TRUE
  OUTPUT "FALSE
END

TO TRUHE.PRUEF
  PRINTL [[SIE IST SCHOEN GEMACHT]
    [UND SICHERLICH EIN KLEINES VER-
    MOEGEN WERT] [AUF DEM DECKEL]
    [IST EIN WINZIGER SCHAEDEL
    EINGESCHNITZT]]
END

TO SCHWERT.PRUEF
  IF HIER? "SCHWERT THEN PRINT [ES IST
    AUS STAHL] ELSE PRINT [ICH SEHE
    KEIN SCHWERT]
END
```

Der Spieler braucht das Schwert, um die Schlange zu töten und somit sein Spielerleben zu retten.

```
TO TOETEN :IT
  IF :IT = "SCHLANGE THEN
    TOETEN.SCHLANGE STOP
  PRINT [DAS GEHT NICHT!]
END

TO TOETEN.SCHLANGE
  IF NOT MEMBER? "SCHLANGE.
    GREIFTAN :GEFAHREN THEN PRINT
    [ICH SEHE KEINE SCHLANGE] STOP
  IF MEMBER? "SCHWERT :INVENTAR
    THEN SCHLANGE.STIRBT ELSE
    SCHLANGE.TOETET
END

TO SCHLANGE.STIRBT
  PRINT [DIE SCHLANGE STIRBT]
  MAKE "GEFAHREN []
END
```



```
TO SCHLANGE.TOETET
  PRINTL [[DU HAST KEINE WAFFEN] [MIT
    DENEN DU SIE TOETEN KANNST]
    [ABER DU HAST SIE JETZT
    ERZUERNT] [SIE BEISST DICH!] [DU
    KRUEMMST DICH UND FAELLST ZU
    BODEN]]
  TOT
END
```

Die Prozedur TOT richtet sich an jene Spieler, die an Wiedergeburt glauben! Wenn Sie, nachdem Sie Ihr "Spielerleben" verloren haben, etwas anderes als START eingeben, wird Sie der Computer daran erinnern, daß Sie eigentlich tot sind!

Der hilfreiche Geist

```
TO TOT
  PRINT [DU BIST TOT!]
  PRINT1 "?
  MAKE "INPUT BEFEHL
  IF ( :INPUT = "START) THEN START
    STOP
  PRINT [LASS ES SEIN!]
  TOT
END

TO BEFEHL
  MAKE "INP REQUEST
  IF :INP =[] THEN PRINT1 "? OUTPUT
    BEFEHL
  OUTPUT FIRST :INP
END
```

Beim Reiben des Ringes erscheint ein Geist:

```
TO REIBEN :OBJ
  IF :OBJ = "RING THEN REIBEN.RING
    STOP
  PRINT [ER STRAHLT JETZT MEHR ALS
    VORHER]
END

TO REIBEN.RING
  IF HIER! "RING THEN GEIST ELSE PRINT
    [ICH SEHE KEINEN RING]
END
```

Der Geist bietet dem Spieler an, ihn nach Hause zu bringen. Wird diese Einladung abgelehnt, bläst ihn ein starker Wind in einen Raum im östlichen Höhlenteil:

```
TO GEIST
  PRINTL [[EIN GEIST ERSCHEINT UND
    FRAGT:] ["WILLST DU NACH
    HAUSE?"]]
  PRINT1 "?
  MAKE "ANT FIRST BEFEHL
  IF ANYOF :ANT = "JA :ANT = "J THEN
    ZURUECK ELSE BLASEN
END

TO ZURUECK
  PRINT [ENDLICH DAHEIM]
  IF MEMBER? "ZEPTER :INVENTAR THEN
    PRINT [GLUECKWUNSCH, ZEPTER
    GEFUNDEN!] ELSE PRINTL [[NA,
    WENIGSTENS BIST DU MIT DEM
    LEBEN]
    [DAVONGEKOMMEN]]
END

TO BLASEN
  PRINT [HIER WEHT EIN HEFTIGER
    STURM]
  PRINT"
  MOVE1 (6 + (RANDOM 5))
END
```

Die Truhe läßt sich öffnen. In ihr befindet sich eine giftige Spinne. Der Schädel auf dem Deckel ist eine Warnung, sie nicht zu öffnen. Doch manche Leute werden nie klug!

```
TO OEFFNEN :OBJ
  IF :OBJ = "TRUHE THEN
    OEFFNEN.TRUHE ELSE PRINT
    [DU KANNST SIE NICHT OEFFNEN]
END

TO OEFFNEN.TRUHE
  PRINTL [[IN DER TRUHE IST EINE]
    [GIFTIGE SPINNE] [SIE BEISST DICH]]
  TOT
END
```

Nachstehend eine Übersicht aller im Spiel verwendeter Substantive:

```
TO SCHWERT
  OUTPUT "SCHWERT
END

TO TRUHE
  OUTPUT "TRUHE
END

TO ZEPTER
  OUTPUT "ZEPTER
END

TO RING
  OUTPUT "RING
END

TO SCHLANGE
  OUTPUT "SCHLANGE
END
```

Soll der erreichte Spielstand zwecks späterer Fortsetzung gespeichert werden, gibt man SAVE „ADVENTURE ein. Durch Eingabe von READ „ADVENTURE wird das Spiel geladen.

LOGO eignet sich unter verschiedenen Gesichtspunkten ideal zum Programmieren von Abenteuerspielen. Ein Problem gibt es jedoch: Bei den zur Zeit verfügbaren LOGO-Versionen steht nicht genügend freier Speicherplatz zur Verfügung. Dieses Spiel paßt gerade noch in den Speicher des Commodore 64. Jedwede Veränderungen bedeuten, daß man Kompromisse schließen muß, welche Worte bestehen bleiben und welche entfernt werden sollen.

Übung macht auch hier den Meister

Dieses Sprichwort hat im Umgang mit Computern seine Gültigkeit. Auf dieser Seite finden Sie einige Übungen und zur Überprüfung Ihrer Antworten auch gleich die jeweiligen Lösungen.

1. Schreiben Sie ein Programm, das die Eingabe von zwei Zahlen über die Tastatur gestattet. Danach sollen die beiden Zahlen addiert und das Ergebnis auf dem Bildschirm ausgegeben werden.

```
100 REM UEBUNG 1
200 INPUT "GEBEN SIE EINE ZAHL EIN";A
300 INPUT "GEBEN SIE EINE ANDERE
    ZAHL EIN";B
400 LET C=A+B
500 PRINT "DIE SUMME DER ZAHLEN
    IST ";C
```

2. Ordnen Sie zwei Worte zwei verschiedenen String-Variablen zu und erstellen Sie dann eine dritte String-Variable, in der die beiden Originalworte zusammengefaßt werden.

```
100 REM UEBUNG 2
200 LET A$="ERSTES WORT,"
300 LET B$="ZWEITES WORT"
400 LET C$=A$+B$
500 PRINT C$
```

3. Schreiben Sie ein Programm, das die Eingabe jedes beliebigen Wortes über die Tastatur zuläßt. Lassen Sie dann mit der Meldung "DAS WORT; DAS SIE EINGEGEBEN HABEN; HAT * ZEICHEN" (* steht für die Zahl) die Anzahl der Zeichen in der String-Variablen ausgeben.

```
100 REM UEBUNG 3
200 INPUT "GEBEN SIE EIN WORT EIN"
    ;W$
300 LET L=LEN(W$)
400 PRINT "DAS EINGEGEBENE WORT
    HAT ";L;" ZEICHEN"
```

4. Schreiben Sie ein Programm, das die Eingabe eines einzelnen Zeichens über die Tastatur zuläßt und Ihnen dann den entsprechenden ASCII-Wert des eingegebenen Zeichens dezimal mitteilt.

```
100 REM UEBUNG 4
200 PRINT "DRUECKEN SIE EINE TASTE"
300 FOR C=0 TO 1 STEP 0
400 LET A$=INKEY$
500 IF A$ <> " " THEN LET C =2
600 NEXT C
700 PRINT "ASCII-WERT VON ";A$;"
    IST "; ASC(A$)
```

Beim Spectrum müssen Sie Zeile 700 wie folgt ändern:

```
700 PRINT "ASCII-WERT VON ";A$;"
    IST "; CODE(A$)
```

5. Schreiben Sie ein Programm, das Sie mit der Meldung "GEBEN SIE EIN WORT EIN" zur Eingabe auffordert und dann mit "DER LETZTE BUCHSTABE DES WORTES WAR EIN *" (*steht für einen Buchstaben) antwortet.

```
100 REM UEBUNG 5
200 INPUT "GEBEN SIE EIN WORT EIN";W$
300 LET L$=RIGHT$(W$,1)
400 PRINT "DAS LETZTE ZEICHEN DES
    WORTES IST EIN ";L$
```

Beim Spectrum sieht dies so aus:

```
100 REM UEBUNG 5
200 INPUT "GEBEN SIE EIN WORT EIN";W$
250 LET N=LEN(W$)
300 LET L$=W$(N)
400 PRINT "DAS LETZTE ZEICHEN DES
    WORTES IST EIN ";L$
```

6. Schreiben Sie ein Programm, das Sie mit der Meldung "GEBEN SIE DEN NAMEN EINER PERSON EIN" zur Eingabe auffordert und dann mit der Meldung "DIE LEERSTELLE WAR DAS *TE ZEICHEN" antwortet.

```
100 REM UEBUNG 6
200 PRINT "GEBEN SIE EINEN NAMEN IN
    FOLGENDER FORM EIN:"
300 PRINT "VORNAME FAMILIENNAME"
400 "Z.B. KARIN BOBEY"
500 INPUT "NAME ";N$
600 LET S=0:LET L=LEN(N$)
700 FOR P=1 TO L
800 IF MID$(N$,P,1)=" " THEN LET S=P
900 NEXT P
950 PRINT "DIE LEERSTELLE IST DAS ";S;
    "*TE ZEICHEN"
```

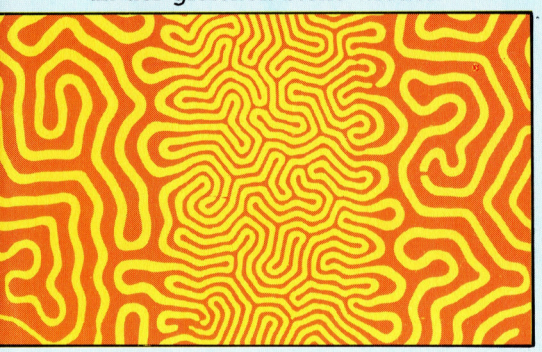
Beim Spectrum ersetzen Sie Zeile 800 wie folgt:

```
800 IF N$(P)=" " THEN LET S=P
```

Fachwörter von A bis Z

Break = Unterbrechung
Die Ausführung eines Programms läßt sich im allgemeinen durch Betätigung einer Sondertaste (z. B. STOP, BREAK, ESCAPE) unterbrechen. Dabei werden alle Informationen über den Stand der Programmbearbeitung in einem reservierten Speicherbereich abgelegt, zum Beispiel in welcher Zeile unterbrochen wurde oder welchen Inhalt Schleifenzähler und Unterprogrammregister hatten. Dadurch ist gewährleistet, daß das Programm ohne Datenverlust oder -veränderung wieder gestartet werden kann.

Solche Unterbrechungen sind beispielsweise bei der Fehlersuche nützlich. Sie unterbrechen an einer kritischen Stelle, schauen sich den gegenwärtigen Variablenwert an, verändern ihn bei Bedarf und starten an der gleichen Stelle wieder.



Magnetfeld-Darstellung

Bubble Memory = Blasenspeicher
In Magnetblasenspeichern wird die Information durch Magnetfelder dargestellt, also nicht durch elektrische Spannungen oder Ladungen wie beim Halbleiter-RAM. Anders als bei Magnetmedien wie Band oder Diskette gibt es beim Blasenspeicher aber keinerlei mechanische Bewegung. Informationsträger ist eine dünne Magnetschicht auf einem Granat-Kristall, in der eine Vielzahl von winzigen „Blasen" mit gegensätzlichen Magnetisierungsrichtungen (entsprechend der logischen Null bzw. Eins) gebildet wird.

Die einzelnen Blasen sind in Form von Schleifen miteinander verbunden und werden durch ein äußeres Magnetfeld in Bewegung gehalten. Diese Speicherschleifen werden über eine zentrale Zugriffsschleife sequentiell abgefragt bzw. beschrieben. Das geht zwar schneller als bei Band oder Platte, jedoch langsamer als bei direktem Speicherzugriff. Demgegenüber hat der Blasenspeicher jedoch den Vorteil, daß die Daten auch bei Stromausfall erhalten bleiben.

Hier werden einzelne Fachausdrücke eingehend behandelt. Da bei der Kommunikation mit dem Computer meist die englische Sprache verwendet wird, werden hier zunächst die englischen Begriffe genannt, dann die deutsche Übersetzung. In den Gesamtindex werden sowohl deutsche als auch englische Stichwörter aufgenommen, damit Sie es leichter haben, das von Ihnen Gesuchte zu finden.

Bubble Sort = Bubble Sort
Der „Bubble Sort" ist ein Sortieralgorithmus, der ein Datenfeld alphabetisch oder nach der Größe ordnet. Das Verfahren ist einfach zu verstehen. Die Arbeitsweise ist jedoch vergleichsweise unwirtschaftlich. Der Name rührt daher, daß beim wiederholten Durchgehen des Feldes die Daten wie Luftblasen (bubbles) zu dem für sie bestimmten Platz im Feld aufsteigen.

Buffer = Buffer
Ein Hardware-Buffer gleicht die elektrischen Spannungsdifferenzen aus, die zwischen den verschiedenen Hardware-Komponenten des Computers entstehen. Die meisten Peripheriegeräte sind in der Lage, mehr Informationen zu verarbeiten, als der Microprozessor in einem bestimmten Zeitraum weiterleiten kann. Ein spezieller Buffer-Chip sorgt dabei für die Verstärkung des digitalen Signals.

Der Software-Buffer gleicht dagegen die Übertragungsdifferenzen zwischen dem Betriebssystem und den anderen Systemkomponenten aus. Dieser Buffer ist im allgemeinen Teil des RAM-Bereiches untergebracht. Er kann zum Beispiel von einem Peripheriegerät mittels einer entsprechend hohen Übertragungsgeschwindigkeit mit Daten gefüllt und von einem anderen, das nur eine niedrigere Rate verarbeiten kann, geleert werden. Buffer dieser Art werden zwischen Prozessor und Tastatur, Diskettenstation und Drukker eingesetzt.

Bei der Textverarbeitung können Sie die Arbeitsgeschwindigkeit mit einem großen Buffer zwischen Rechner und Drucker erheblich vergrößern. Wenn Sie den Druckbefehl geben, lädt der Rechner die fertige Seite in den Buffer. Dieser wird dann vom Drucker langsam geleert, während Sie bereits die nächste Seite eintippen, ohne warten zu müssen. Kleinere Buffereinheiten haben eine Kapazität bis zu vier KByte, was für ein Schriftstück mit 700 Wörtern ausreicht.

Bus = Bus
Ein Bus ist ein elektrischer Übertragungskanal für Informationen innerhalb eines Rechnersystems, meist in Form einer Anzahl von Leiterbahnen auf der Platine. Der Systembus ist über den Bus-Erweiterungsstecker von außen erreichbar. Intern verbinden drei wichtige Busse die CPU mit den übrigen Komponenten: Der „Adreßbus" überträgt Speicher- und Geräteadressen, der „Datenbus" die eigentlichen Informationen, und über den „Steuerbus" werden Signale zur Steuerung und Synchronisation der verschiedenen Systembausteine ausgetauscht.

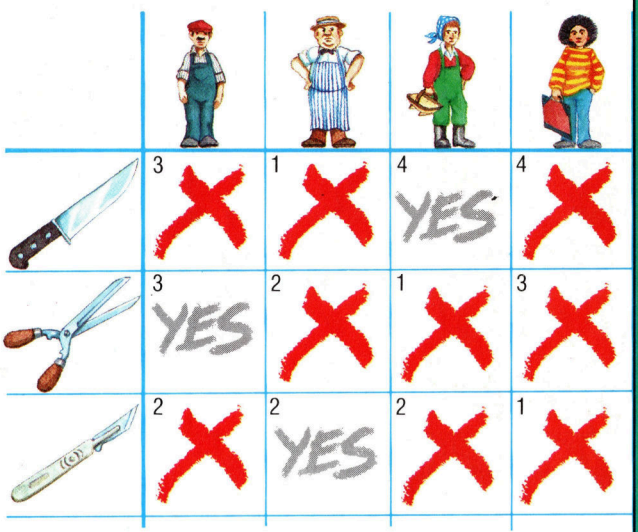
Bildnachweise
617, 619, 640: Steve Cross
618, 633, 636, 637: Kevin Jones
620, 622, 623: Ian McKinnell
621: Collins/Duncan Smith
628: Acorn
629, 630, 631: Chris Stevens
638: Liz Heaney
641: Record Mirror
643: Liz Dixon
U3: New Scientist

++ Vorschau +++ Vorschau +++ Vorschau +++

computer kurs Heft 24

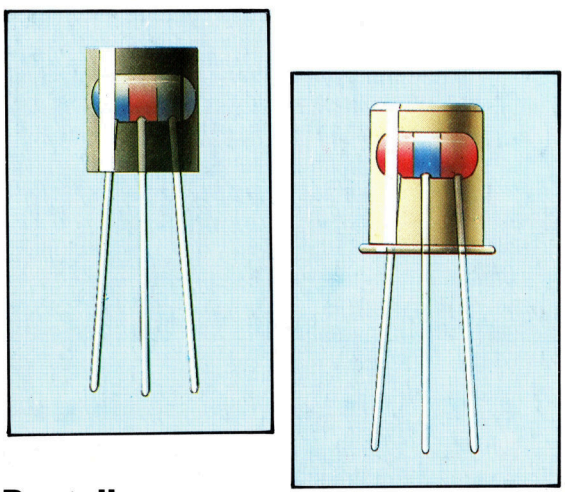
Klänge mit dem Sequenzer

beschreibt die Serie „Computer und Musik". Der Sequenzer hat das Musikschaffen auf der Bühne wie im Studio stark beeinflußt.



Mordfall-Untersuchung

Listenverarbeitung im LOGO-Kurs: Als Beispiel dient die Untersuchung in einem Mordfall, in dem es den Täter zu finden gilt.



Bauteile

Jedes elektronische Gerät besteht aus einer Reihe von Bauteilen. Diesmal geht es um Transistoren und Widerstände.

+++ Spurensicherung +++ Fachwörter
+++ Software: Lageranalyse +++
Maschinenprogramme +++ BASIC:
Sprachvergleich +++ Roboter-Serie +++
Micro Advance 86 +++ Firma Imagine